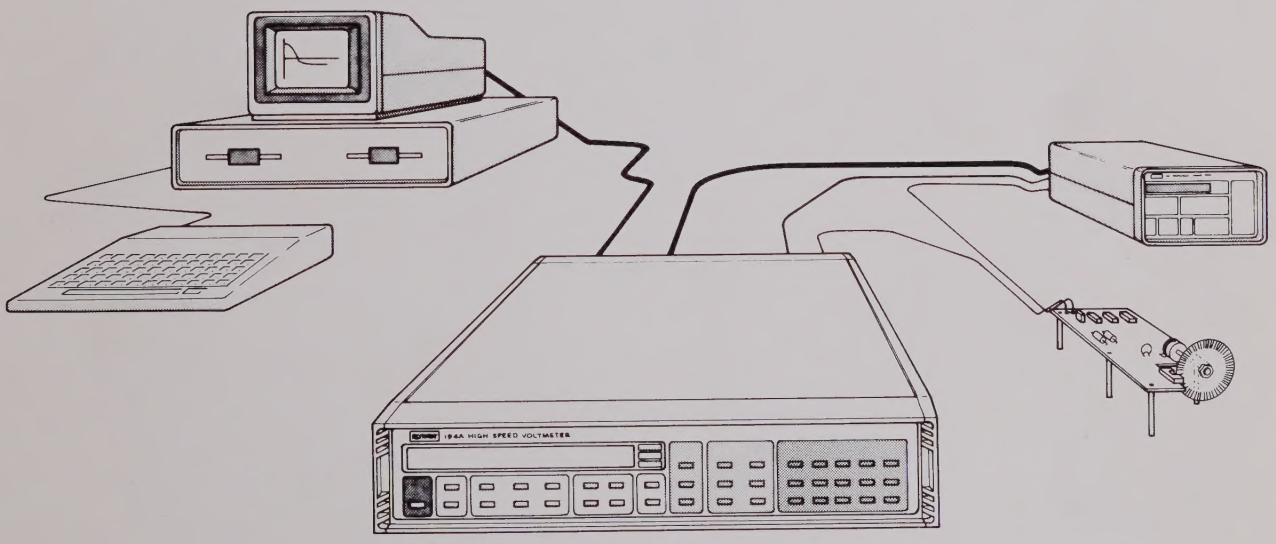
KEITHLEY

Model 194A
High Speed Voltmeter
IEEE-488 Programming Guide



Contains Programming Tutorial, Simple Example Programs and System Application Programs

Publication Date: March 1987
Document Number: 194A-903-01 Rev. A

Digitized by the Internet Archive
in 2024

https://archive.org/details/manualsplus_02253

IEEE-488 Programming Guide
Model 194A
High Speed Voltmeter

©1987, Keithley Instruments, Inc.
Instrument Division
Cleveland, Ohio, U.S.A.
Document Number: 194A-903-01 Rev. A

TABLE OF CONTENTS

About This Programming Guide 1

Computer Programming Syntax 3

Single Channel Programming 7

Dual Channel Programming 11

Single and Continuous Trigger Arming 17

SRQ and Error Handling 21

Binary Data Transfer 29

Reading Buffer Operation 37

Using Translator 41

Using External Sample Triggering 47

Generating Bus Time Information 55

Appendix A - Device-Dependent Commands A-1

Appendix B - Data Formats B-1

Appendix C - Status Words and SRQ Format C-1

Appendix D - Front Panel Modes and Equivalent IEEE-488 Commands D-1

Appendix E - Data Transmission and NRFD Hold-off Times E-1

ABOUT THIS PROGRAMMING GUIDE

This programming guide is intended as a tutorial on a number of programming concepts associated with the Model 194A. It is not intended to replace the IEEE-488 programming section of the Instruction Manual, which should be consulted for detailed programming information.

Programming concepts discussed in this guide include:

- Programming a single-channel unit and obtaining data.
- Programming a dual-channel unit and requesting data.
- Single/continuous arming and triggering.
- Using binary data transfer to optimize speed.
- Using SRQ and error word to minimize controller overhead.
- Buffer operation, including use of buffer pointers.
- Using Translator to increase readability and emulate other instrumentation.

In addition to these fundamental concepts, applications programs for external sample triggering, plotting data, and trigger performance and timing are also included. Finally, reference information such as device-dependent commands, data formats, and error words is also supplied.

COMPUTER PROGRAMMING SYNTAX

All the example programs in this guide are written in Hewlett-Packard BASIC 4.0, which is used on the Series 200 and 300 computers. This programming language was chosen for these examples because of its ease of understanding, as well as its powerful subset of commands and statements controlling the IEEE-488 bus.

Most BASIC 4.0 statements are very similar to those used in other BASIC languages; however, there are some statements associated with IEEE-488 bus I/O that require discussion. The following provides a brief overview of some of the statements you will encounter in the various programs in this guide. Refer to the BASIC 4.0 Reference for more detailed information.

OUTPUT

OUTPUT allows you to send a variable or string over the bus, and it may take on one of several forms. A typical syntax is:

```
OUTPUT 709;"F1X"
```

In this instance, 7 specifies the interface select code, while 09 is the primary address of the instrument. F1X is a literal string that will be transmitted to the instrument when the statement is executed.

ENTER

ENTER performs the opposite function, allowing the transmission of data from the instrument to the computer. Typical syntax for this statement is;

```
ENTER 709; A$
```

Again, 7 specifies the interface select code, while 09 represents the primary address of the instrument. A$ is a string variable into which incoming data is placed. Keep in mind that A$ must be dimensioned large enough to hold all expected characters. As an alternative, the ENTER statement can directly input the data into a numeric variable, assuming that the proper instrument data format is used.

SPOLL

SPOLL serial polls the instrument and places its status byte in a numeric variable as follows:

```
S = SPOLL(709)
```

Here again, 709 represents the interface select code and primary address of the instrument. S is a numeric variable which will contain the decimal value of the status byte after the state-

ment is executed. The status byte can then be checked by ANDing or BITing S to determine the status of the required bits. These bits can yield certain information such as reading overflow and ready for a new command.

BIT

BIT can be used to determine the bit status of numeric variables:

```
A = BIT(S,5)
```

In this example, the status of bit 5 of the variable S is placed into A.

TRIGGER

TRIGGER sends a GET (Group Execute Trigger) to a device as follows:

```
TRIGGER 709
```

A GET is one of many methods that can be used to initiate a Model 194A measurement. In order to use this form of triggering, the GET trigger mode must be programmed, and the A/D converter must be armed. Note that programming a trigger mode over the bus automatically arms the A/D converter.

ASSIGN

ASSIGN is used to assign an I/O path, as in the following example:

```
ASSIGN @D194 TO 709
```

Following ASSIGN execution, the attribute D194 will be associated with the interface code and primary address. Another form of ASSIGN is used to define the I/O buffer:

```
ASSIGN @Buf TO BUFFER A$
```

TRANSFER

TRANSFER is used to perform a rapid I/O data transfer sequence. Usually, TRANSFER is used with the binary dump feature of the Model 194A for most rapid data transfer. Using this combination, data can be transferred at a rate as high as 90K bytes per second.

Typical syntax for TRANSFER is:

```
TRANSFER @D194 TO @BUF; COUNT 1000,WAIT
```

In this instance, a total of 1000 bytes will be transmitted from the Model 194A to the internal buffer.

STATUS

STATUS allows access to certain computer status registers. Most often, this statement is used to either determine if an SRQ has occurred, or clear an SRQ interrupt. Typical syntax is:

```
STATUS 7,7;S
```

Here, 7 is the interface select code, and the second 7 is the status register being accessed. The value of the status register is placed in the variable S when the statement is executed.

ON INTR and ENABLE INTR

These two statements work together to enable interrupts and to tell the computer where to go for next line execution. A typical ON INTR statement is:

```
ON INTR 7 GOSUB 4000
```

In this case, when the computer detects an IEEE-488 interrupt, it begins executing the subroutine at line 4000.

SRQ is the most often used type of interrupt for the IEEE-488 bus. Using BASIC 4.0, SRQ interrupts are enabled as follows:

```
ENABLE INTR 7;2
```

Once enabled, an SRQ will cause the computer to branch to the line number specified by the ON INTR statement. Typically, an error handling subroutine will be incorporated at that point to determine the reason for the SRQ and to handle any other required tasks.

SINGLE CHANNEL PROGRAMMING

PROGRAMMING CONCEPTS

- Sending command strings to the Model 194A.
- Obtaining data and displaying it on the computer CRT.

PROGRAMMING CONSIDERATIONS

At the very least, it will be necessary to send commands to the instrument to set it up as desired, and to request data back from the unit. The program below demonstrates the basic process for doing so. Note that if you have a dual-channel unit, you should proceed to Program 2.

PROGRAM OPERATION

Program 1 below demonstrates fundamental programming techniques for a single-channel unit. Figure 1 is a flowchart of the program.

Before we can program the Model 194A, we must place the unit in remote as follows:

```
30 REMOTE 709
```

Next a prompt for the user to enter the desired command is displayed by the following program statements:

```
40 PRINT ''ENTER COMMAND''
50 INPUT COMMAND$
```

Following execution of these two lines, the command string resides in a string variable called Command$. Now we can send the command string to the instrument with the "X" character appended to the string to force command execution:

```
60 OUTPUT 709;COMMAND$&''X''
```

Since sending any A/D (Analog-to-Digital) converter programming command to the instrument will disarm the A/D converter (see Table 1), you should always arm the A/D converter before requesting a reading. In this particular program, the A/D converter is armed by sending the T26 command in the following manner:

```
70 OUTPUT 709;''T26X''
```

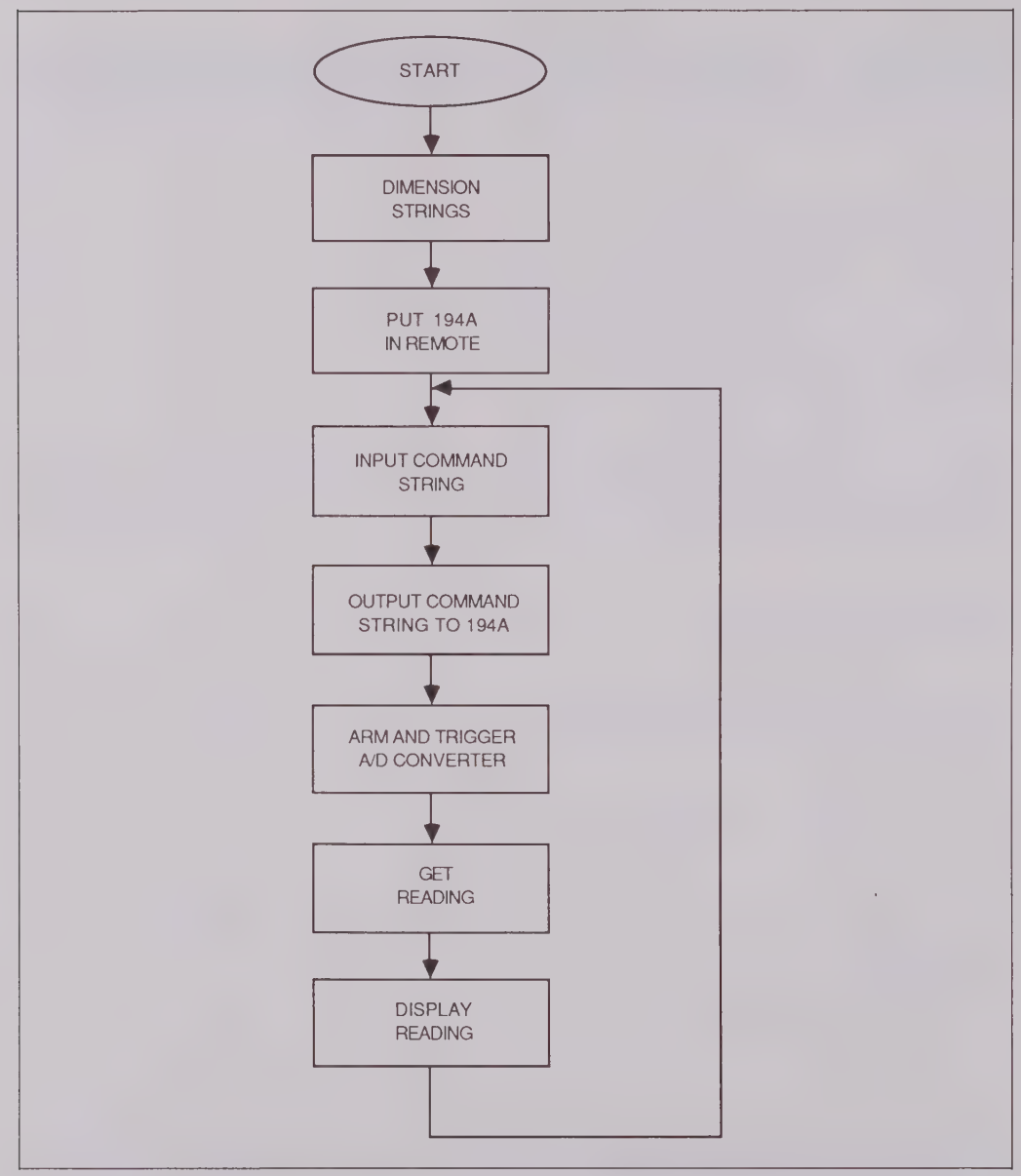


Figure 1. Single Channel Operation Flowchart

Note that the immediate trigger mode is used in this instance, so it is not necessary to trigger the unit to process a measurement; however, with other trigger modes, it will be necessary to trigger the unit before the unit will collect samples. Once those samples are stored and processed into a reading, the data is requested and displayed on the computer CRT as follows (see Figure 2 for ASCII data format):

```
80 ENTER 709;Reading$
90 PRINT Reading$
```

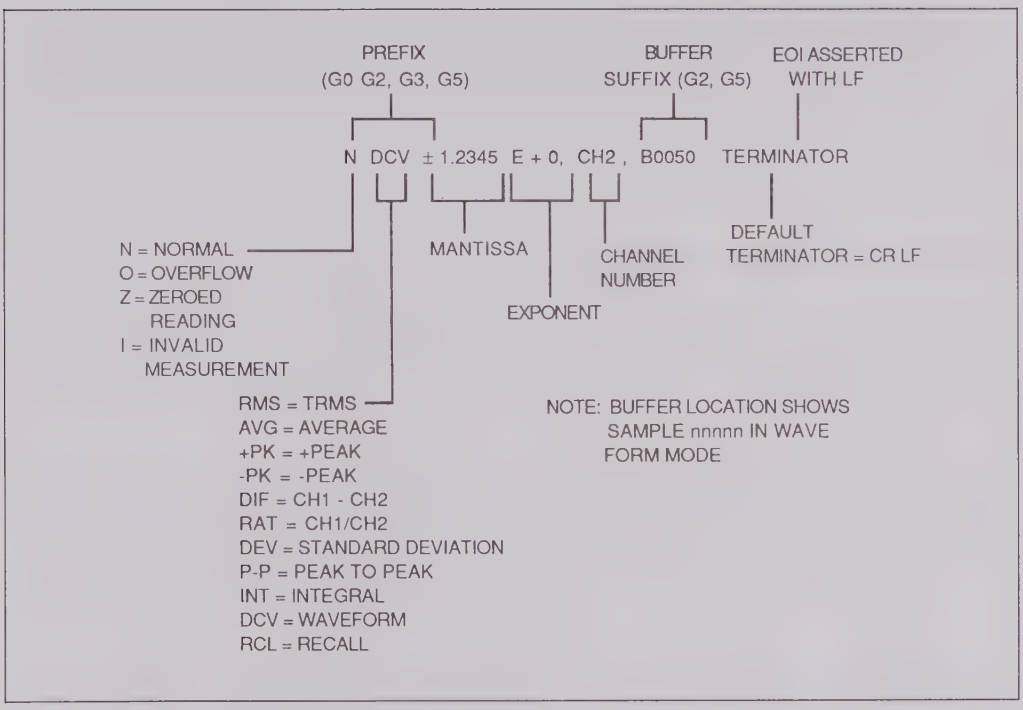


Figure 2. ASCII Data Format

Table 1. Commands That Disarm the A/D Converter

Command	Description
A	Recall setup
F	Function
I	Coupling
J	Self test
N	Number of samples
P	Filter
R	Range
S	Rate
T	Trigger
W	Delay
Z	Zero

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Run the program by pressing the RUN key. The computer will prompt you to type in your command:

```
ENTER COMMAND?
```

3. Type in the desired command string and then press ENTER. Note that is not necessary to add the "X" character as it is automatically added by the program. For example, to program the average function and select autoranging, type in F1R0 and press ENTER.
4. Once the measurement is completed and the reading is processed, the data string will appear on the computer display. A typical example is:

```
NDCV+1.2500E+0;CH1
```

ACQUIRING A WAVEFORM USING ASCII DATA TRANSFER

The Model 194A can be used to capture a waveform and then send the data as a series of readings representing the individual samples. Each individual data string will represent the voltage of that particular sample.

To use the Model 194A in this manner, program the instrument for the waveform (F0) function, and then repeatedly request readings until all samples have been transferred. For example, to collect 100 samples and read them into a computer, modify Program 1 as follows:

```
75 FOR I = 1 TO 100          ! Loop 100 times.
95 NEXT I                    ! Loop back for next sample.
```

When the computer prompts you for a command, type in "F0N0,100" (include the quotes because of the comma) and then press ENTER. You should see 100 successive readings appear on the display. Each reading is the numeric value of that particular sample. These samples can also be placed into a BASIC array for further processing, if required.

For a larger number of samples, or for higher speed data transfer, refer to the section in this guide on binary data transfer.

Program 1. Single Channel Operation

```
10    DIM Reading$[50]          ! Dimension reading string
20    DIM Command$[20]          ! Dimension command string
30    REMOTE 709                ! Put 194 in remote
40    PRINT "ENTER COMMAND"     ! Prompt for device dependent command
50    INPUT Command$            ! Input command string
60    OUTPUT 709;Command$&"X"   ! Output command to 194A
70    OUTPUT 709;"T26X"         ! Arm A/D converter and trigger immediately
80    ENTER 709;Reading$        ! Get reading
90    PRINT Reading$            ! Print reading
100   GOTO 40
110   END
```

DUAL-CHANNEL PROGRAMMING

PROGRAMMING CONCEPTS

- Basic command programming of a dual-channel Model 194A.
- Obtaining data from a dual-channel Model 194A.
- Demonstrating that both channels share a single IEEE-488 output.
- How to turn off an unused channel for maximum speed and to ensure that data comes only from the desired channel.

PROGRAMMING CONSIDERATIONS

Programming a dual-channel unit is somewhat more complex than a single-channel unit, even for the most basic measurements. In particular, there are several important aspects to keep in mind:

1. Before sending programming commands to the instrument, select the desired channel by sending C1X for channel 1, or C2X for channel 2. Doing so will ensure that the commands you send will affect the desired channel. Note that some commands such as data format (G), SRQ (M), and terminator (Y) affect both channels, so it is not necessary to select the channel before using them.
2. The two channels share a common IEEE-488 output and reading buffer. As a result, there is no way to dictate which channel has the current output when both channels are running simultaneously. Thus, it will be necessary for you use the channel suffix in the data string (Figure 2) to determine the source of the data. Alternatively, you can use the reading buffer for storage and request data later.
3. If one channel is not being used, it should be disabled by sending a trigger command for a trigger stimulus that will not occur. For example, if you are not using external triggering, send the command string C1XT7X to disarm channel 1, or send C2XT7X to disable channel 2. Turning off an unused channel will accomplish two things: (1) maximize throughput for the channel being used, and (2) make sure that data comes only from the channel in use.
4. Another way to keep the reading from both channels separated is to use the F command to re-calculate a reading. Even if a reading has already been calculated, you can force another calculation for the desired channel to place the result in the reading buffer. For example, send C2F1X to compute the average of the channel 2 measurement and place the results in the reading buffer. Keep in mind that the other channel must not be running fast enough to overwrite this data between the time it is calculated and the time it is requested.

PROGRAM OPERATION

Program 2 below demonstrates dual-channel programming. Figure 3 is a flowchart showing general program flow.

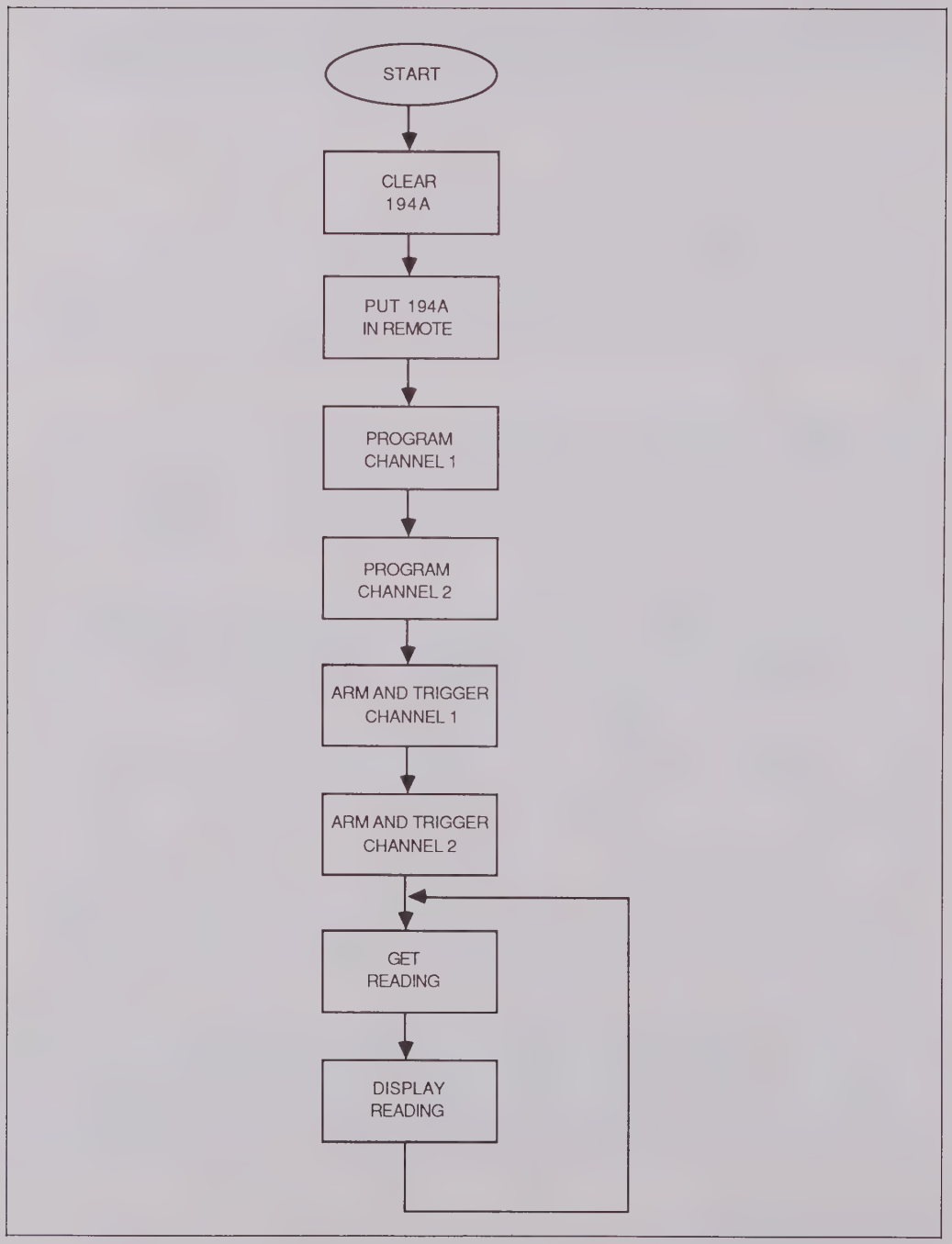


Figure 3. Dual Channel Operation Flowchart

Usually, an instrument is sent a command at the start of a program to return it to some default state. In the program below, this operation is performed by sending an SDC (Selected Device Clear) as follows:

```
20 CLEAR 709
```

This statement instructs the Model 194A to assume its power-up (setup 1) configuration.

Next, we must make certain that the instrument is in remote before sending programming commands as follows:

```
30 REMOTE 709
```

Now we can program channel 1 and channel 2 as required. For channel 1, we will select the 320mV range (R1) and TRMS function (F2) as follows:

```
40 OUTPUT 709;''C1XR1F2X''
```

In a similar manner, channel 2 is programmed for the 200V range (R4) using the average function (F1):

```
50 OUTPUT 709;''C2XR4F1X''
```

In both cases, you will note that the channel select (C) command is included at the front of the command string containing range and function commands, followed by an execute (X) character to assure proper channel selection. (Note that the X immediately after the C command is not required, but it does no harm).

At this point, we have programmed both channels; however, since the range and function were changed, we must arm both A/D converters. First, for channel 1, we have:

```
60 OUTPUT 709;''C1XT26X''
```

Also, to arm channel 2, we send:

```
70 OUTPUT 709;''C2XT26X''
```

Again, each command string includes the channel selection command prior to the trigger command that arms the A/D converter. Here again, the immediate trigger mode (T26) is used so that no further triggering is necessary to start the A/D converters. With other trigger sources, you will be required to apply the necessary trigger stimuli to start the A/D converters after arming them.

Now that both A/D converters are running and readings are being processed, it is a simple matter to set up a loop to request and display readings:

```
80 ENTER 709;Reading$
90 PRINT Reading$
100 GOTO 80
```

This loop repeats indefinitely, with one reading per loop accessed and displayed.

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Press the computer RUN key to start the program.
3. The program will send the commands to the instrument and then request readings from the Model 194A. Note that data will alternate between channel 1 and channel 2 depending on which channel has the most current reading.

USING ONE CHANNEL

In order to use only one channel, program the other channel for an unused trigger source. For example, to turn off channel 1, modify line 60 as follows:

```
60 OUTPUT 709;''C1XT7X''
```

Similarly, to disable channel 2, modify line 70 to read:

```
70 OUTPUT 709;''C2XT7X''
```

In both instances, the disabled channel is programmed for the external trigger mode. Thus, in order for that channel to remain disabled, no trigger signal can be applied to the trigger input for that channel. Another way to disable a channel is to use T27 which causes only one trigger; no further triggers will occur even with external trigger input noise.

SELECTING CHANNEL OUTPUT

As discussed previously, you can dictate which channel has the current output by programming the desired channel for a particular function (such as average), and then immediately requesting a reading. In order to demonstrate this process, eliminate lines 80-110 from Program 2, and add the following lines:

```
 80 OUTPUT 709; ''C1F1X''      ! Compute  channel 1 reading.
 90 ENTER 709; Reading$        ! Get channel 1 reading.
100 PRINT Reading$             ! Display channel 1 reading.
110 OUTPUT 709; ''C2F1X''      ! Compute  channel 2 reading.
120 ENTER 709; Reading$        ! Get channel 2 reading.
130 PRINT Reading$             ! Display channel 2 reading.
140 GOTO 80                    ! Repeat
150 END
```

With these modifications, the computer display will alternate between channel 1 and channel 2 readings. Note, however, that one channel may still overwrite the other if there is a large difference in processing speed between the two channels.

Program 2. Dual-Channel Programming

```
10    DIM Reading$[50]              ! Dimension reading string
20    CLEAR 709                     ! Return the 194A to default conditions
30    REMOTE 709                    ! Put the 194 into remote mode
40    OUTPUT 709;"C1XR1F2X"         ! Program ch1 for the 320mV range and TRMS
50    OUTPUT 709;"C2XR4F1X"         ! Program ch2 for the 200V range and Average
60    OUTPUT 709;"C1XT26X"          ! Arm channel 1
70    OUTPUT 709;"C2XT26X"          ! Arm channel 2
80    ENTER 709;Reading$            ! Get a reading from the 194A
90    PRINT Reading$                ! Print the reading
100   GOTO 80                       ! Repeat
110   END
```

SINGLE AND CONTINUOUS TRIGGER ARMING

PROGRAMMING CONCEPTS

- Sending the arming command once with continuous arming.
- Sending the arming command before each trigger with single arming.
- Triggering the measurement with single and continuous arming.

PROGRAMMING CONSIDERATIONS

Before the Model 194A will take a measurement, it must be triggered with the stimulus determined by the selected trigger source: immediate, input signal, external, other channel, or an IEEE-488 GET, X, or talk command. However, it will not respond to a trigger unless the A/D converter is first armed and ready for a trigger.

Basically, there are two A/D arming modes: single and continuous. With single arming, the A/D converter must be re-armed before each trigger. However, in continuous, the A/D converter need be armed only once, as the converter is automatically re-armed after each measurement.

Over the IEEE-488 bus, arming takes place when the programming command for that particular trigger mode is received. For example, sending T3X (single arm, trigger on GET), arms the A/D converter. Once armed, the measurement will be initiated when GET (Group Execute Trigger) is received. In this instance, the T3X command must be sent before each trigger to arm the A/D converter. In contrast, if using continuous GET (T2X), you need send the command only once before the first trigger, after which the A/D converter will automatically re-arm itself after each measurement.

To summarize, there are several steps to programming and using triggers:

1. First select the channel to be armed and triggered. For example, to select channel 1, send C1X.
2. Send the command to select the trigger source and arm the A/D converter. For example, to select single, GET, send T3X.
3. Repeat steps 1 and 2 for the other channel, if desired.
4. Apply the necessary trigger stimulus to initiate the measurement. For example, issue the GET command to initiate the measurement.
5. If you have selected a single arming mode, remember to re-arm the converter before each trigger by sending the appropriate command over the bus.

PROGRAM OPERATION

Program 3 below demonstrates single arming using the GET trigger source. Figure 4 is a flowchart of the program.

Once the unit is in remote, we can program the trigger source and arm the channel 1 A/D converter as follows:

```
20 OUTPUT 709;''C1XT3X''
```

In this particular instance we have selected the single arm, GET trigger mode (T3X); after the command is sent, the A/D converter will be armed.

Now that the A/D converter is armed, we can trigger a measurement by sending GET:

```
30 TRIGGER 709
```

The next two lines request a reading from the Model 194A and display a reading on the CRT:

```
40 ENTER 709;Reading$
50 PRINT Reading$
```

Here the incoming data string is stored in Reading$.

One final line necessary to complete the loop is:

```
60 GOTO 20
```

Note that this branch goes back to the line containing the re-arm (T3X) command because a single-arm trigger mode is in effect. If instead we had branched back only to line 30, only one reading would be taken the first time through the loop.

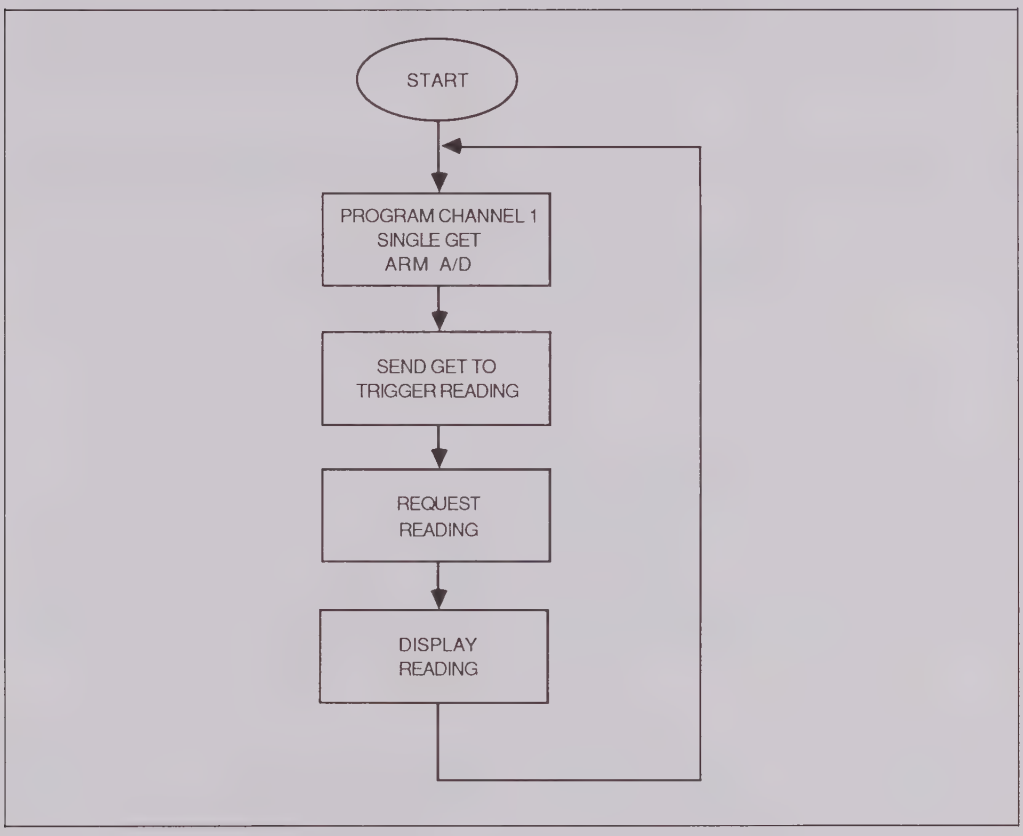


Figure 4. Single Arming Flowchart

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Press the RUN key to execute the program.
3. The program will setup the Model 194A for the single, GET trigger mode, trigger a reading, and display it. A loop is set up to repeat the sequence indefinitely.
4. To demonstrate the importance of re-arming, modify the program as follows:

```
60 GOTO 30                    ! Loop back to trigger.
```

5. Run the modified program and note that only one reading from channel 1 is displayed. With this change, the program now branches back to the trigger statement (line 30) instead of the arming statement (line 20). As a result, only one measurement and reading from channel 1 are processed on the first time through.

CONTINUOUS ARMING

Program 3 can be modified for continuous, GET arming by changing the lines below. Figure 5 shows a flowchart for this modification.

```
20 OUTPUT 709;''C1XT2X''        ! Program continuous arm, trigger on
                                   GET.
60 GOTO 30                      ! Branch back to trigger statement.
```

Upon running this program, you will note that repeated readings appear on the computer display. Now, however, it is not necessary to re-arm the converter each time--merely trigger it--to obtain each reading.

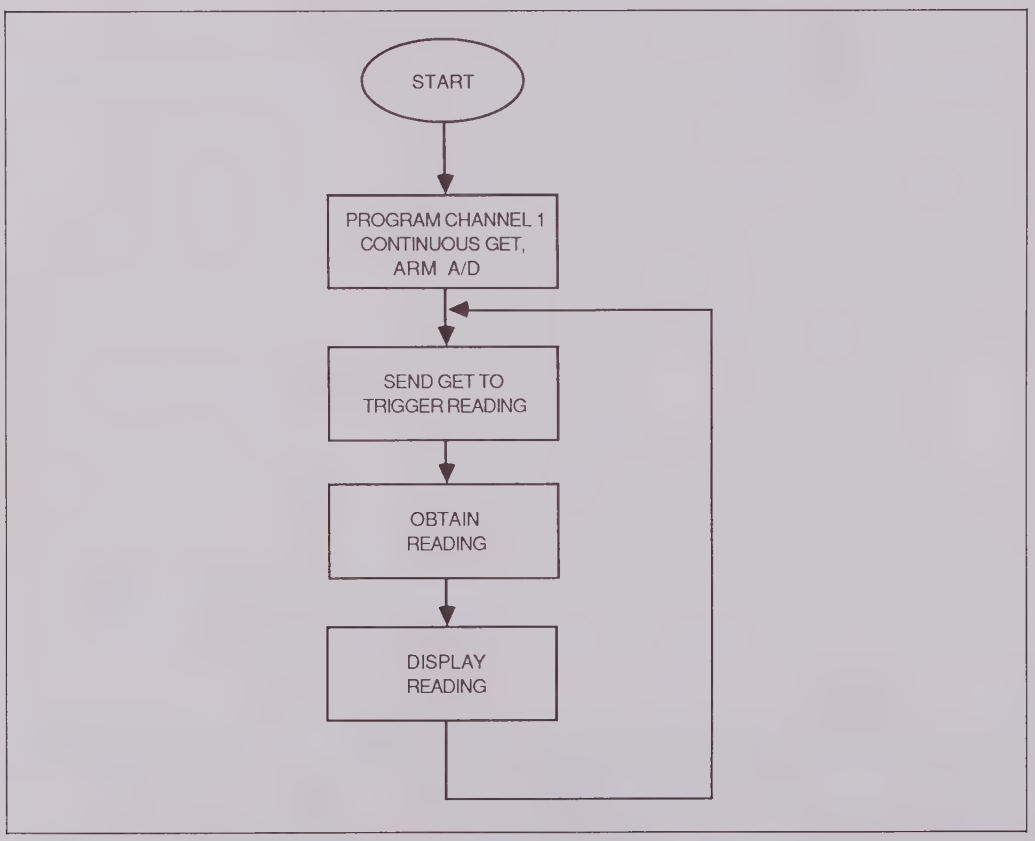


Figure 5. Continuous Arming Flowchart

Program 3. Arming and Triggering

```
10    REMOTE 709                ! Put 194A in remote
20    OUTPUT 709;"C1XT3X"       ! Program channel 1 for single arming with GET
30    TRIGGER 709               ! Trigger the 194A by sending GET
40    ENTER 709;Reading$        ! Take reading from the 194
50    PRINT Reading$            ! Print reading
60    GOTO 20
70    END
```

SRQ AND ERROR HANDLING

PROGRAMMING CONCEPTS

- Programming the instrument to generate an SRQ on reading done and error conditions.
- Servicing the SRQ and determining the exact condition that caused it.
- Taking appropriate action on the SRQ condition.

PROGRAMMING CONSIDERATIONS

The SRQ line on the IEEE-488 bus gives the user a great deal of flexibility in determining if the instrument requires controller servicing. By sending the appropriate command (Table 2), you can program the Model 194A to notify the controller when it requires service for a variety of different conditions including reading overflow, data (buffer full, ½ full, plotter done), front panel button pressed, reading done, ready for command, and a number of error conditions. The instrument can be programmed for multiple SRQ conditions simply by adding up the command values. For example, to program for reading done (M8) and error (M32), M40 would be used.

Once the controller has received the SRQ, it can serial poll the instrument and obtain the status byte to determine whether that instrument did in fact request service, and if so, what conditions caused it to do so (Figure 6). For data and error conditions (bits 1 and 5 respectively), you can check the U1 and U2 status words to determine the exact condition that caused the SRQ, as shown in Figures 7 and 8.

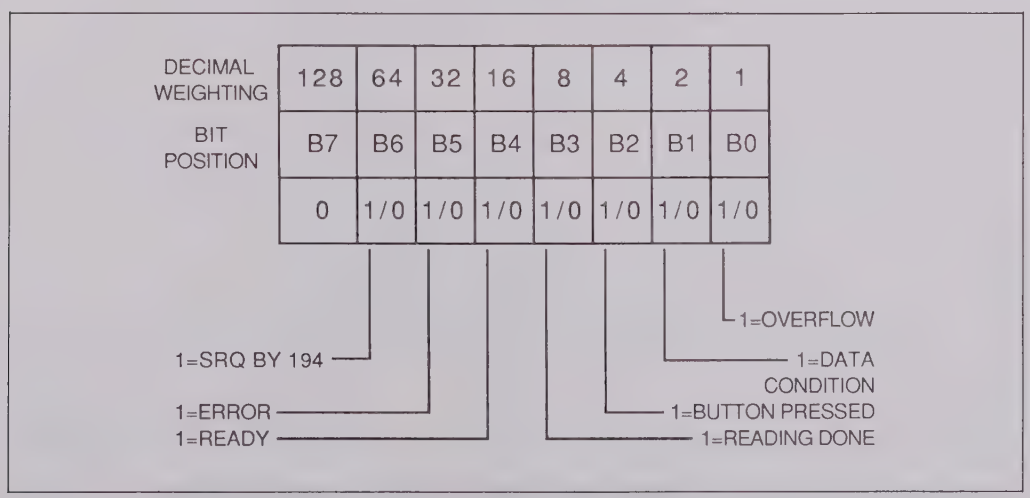


Figure 6. Status Byte Format

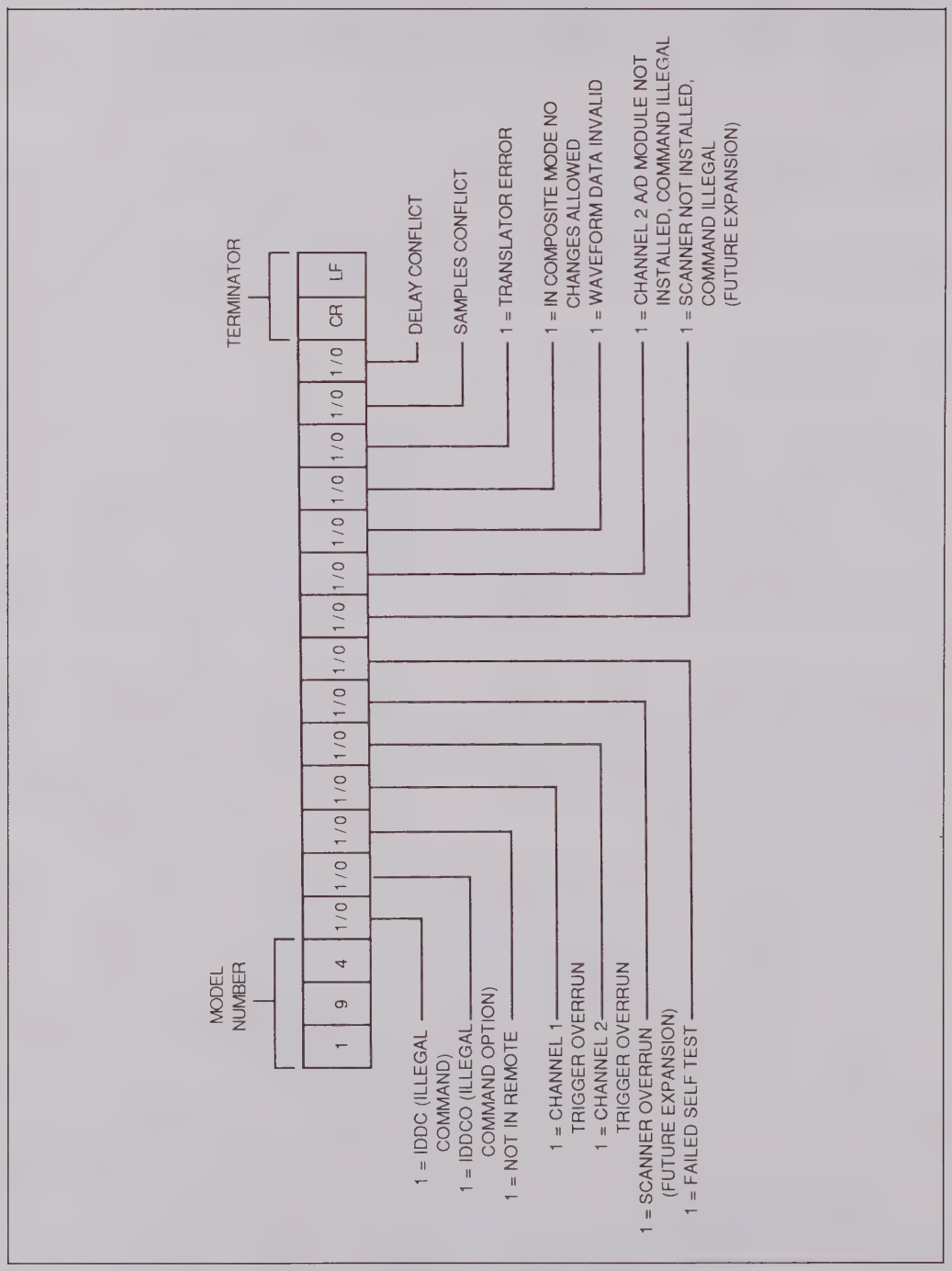

MODEL NUMBER
1
9
4
1/0
TERMINATOR
CR
LF
1 = IDDC (ILLEGAL COMMAND)
1 = IDDCO (ILLEGAL COMMAND OPTION)
1 = NOT IN REMOTE
1 = CHANNEL 1 TRIGGER OVERRUN
1 = CHANNEL 2 TRIGGER OVERRUN
1 = SCANNER OVERRUN (FUTURE EXPANSION)
1 = FAILED SELF TEST
DELAY CONFLICT
SAMPLES CONFLICT
1 = TRANSLATOR ERROR
1 = IN COMPOSITE MODE NO CHANGES ALLOWED
1 = WAVEFORM DATA INVALID
1 = CHANNEL 2 A/D MODULE NOT INSTALLED, COMMAND ILLEGAL
1 = SCANNER NOT INSTALLED, COMMAND ILLEGAL (FUTURE EXPANSION)

Figure 7. U1 Status Word Format (Error Conditions)

Basically, there are two different methods the controller can use to detect the SRQ: interrupt and polling. The interrupt method is by far the more versatile, since it is possible for the controller to perform other tasks until the instrument requests service. With the polling method, the controller must periodically check the status of the SRQ line to determine if the instrument requires service.

Table 2. SRQ Commands

Command*	Description
M0	SRQ disabled
M1	Reading overflow
M2	Data (Buffer full, ½ full, plotter done)
M4	Front panel button pressed
M8	Reading done
M16	Ready for new command
M32	Error (IDDC, IDDCO, not in remote, trigger overrun, channel 2 not installed waveform invalid, composite mode, translator, samples or delay conflict.

*Program multiple SRQ conditions by adding individual command values. For example, send M40X for SRQ on error, and when reading is done.

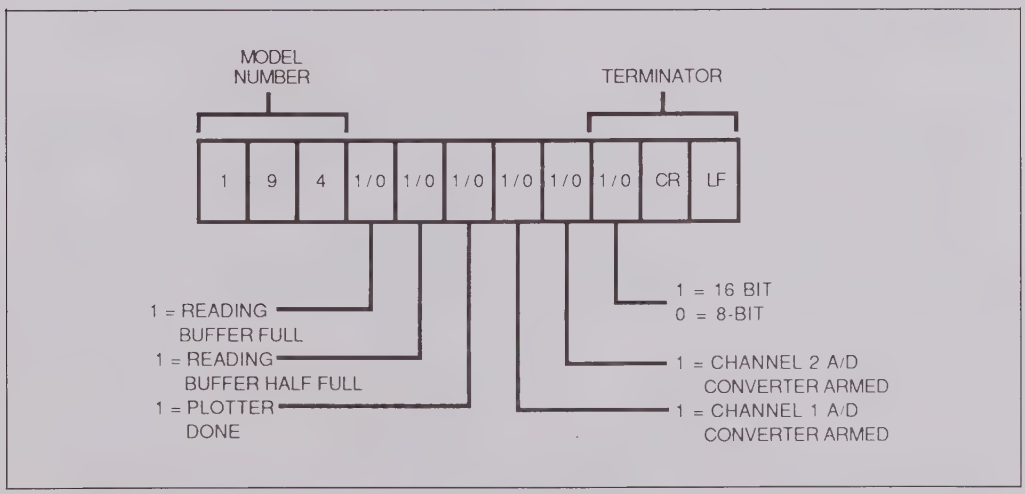


Figure 8. U2 Status Word Format (Data Conditions)

PROGRAM OPERATION

Program 4 below demonstrates interrupt SRQ handling and error status checking. The program can be broken into two sections: initialization and main task, and error handling. Figure 9 gives a flowchart of key sections of the first part of the program, while Figure 10 outlines the error handling routines.

Highlights of the first part of the program include:

1. The Model 194A is returned to default by sending an SDC (line 20).
2. The unit is serial polled in order to clear a pending SRQ (line 30).
3. The service routine that will detect and service the SRQ on an interrupt basis is then defined (lines 70-100).
4. Channel 2 of the instrument is disabled by sending C2T7X (line 120).
5. The user command string is requested and sent to the instrument (lines 130-150).
6. The instrument is programmed to generate an SRQ on reading done or error conditions by sending the command M40X (line 160).
7. The unit is programmed for continuous, GET trigger by sending T2X (line 170). Note that the channel 1 A/D converter will continuously arm in this mode; it will not be necessary to re-arm the converter before every trigger.
8. Since the A/D converter will be armed at this point, it is only necessary to send a GET command to trigger the unit and initiate the measurement (line 180).
9. The user's main task then begins execution. In this instance, the "main task" is a simple incremental counter to indicate that the program is looping (lines 200-220).

When executing, the program will stay in the main task until an SRQ is detected. When an SRQ occurs, the program will service the SRQ as follows:

1. The Model 194A is first serial polled to obtain the status byte (line 250).
2. Next, the state of bit 6 in the status byte is tested to determine if the SRQ was, in fact, generated by the Model 194A (line 270). If this bit is set, the program will continue checking other bits in the status byte. Otherwise, the program returns to the main task. This step is essential only when more than one instrument is on the bus.
3. Bit 5 of the status byte is then checked (line 290) to determine if an error condition occured. If so, the program branches to a subroutine (beginning at line 350) which requests the U1 error word and then tests appropriate bits in the error word to determine the nature of the error. A suitable message is then displayed if an error is found, and the program then halts.
4. If no error is found, the program tests the state of bit 3 in the status byte (line 310). If this bit is set, a reading is ready to be processed, and the program branches to a subroutine to request and display a reading, and trigger a new one (lines 530-570). The subroutine then returns to the user's main task.

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Execute the program by pressing the RUN key.
3. At the command prompt, type in the desired string command. The command string should contain only A/D programming commands (range, function, filter, coupling, sample, and rate). In order to allow sufficient time for the main task to run, choose a sampling rate and number of samples resulting in a measurement duration of several seconds. For example, to program a sampling interval of 1msec and 3000 samples, enter "S0,1E-3N0,3000X" (quotes must be used because of the commas).
4. The computer will then display the incremental counter to show that the main task is running.

5. Once the Model 194A has taken its measurement and processed the reading, the unit will generate an SRQ, and the reading will be displayed.
6. To demonstrate an SRQ-generated error message, type in an illegal command at the command prompt. For example, type in E1X; note that the computer displays the IDDC ERROR because E is an illegal device-dependent command letter.

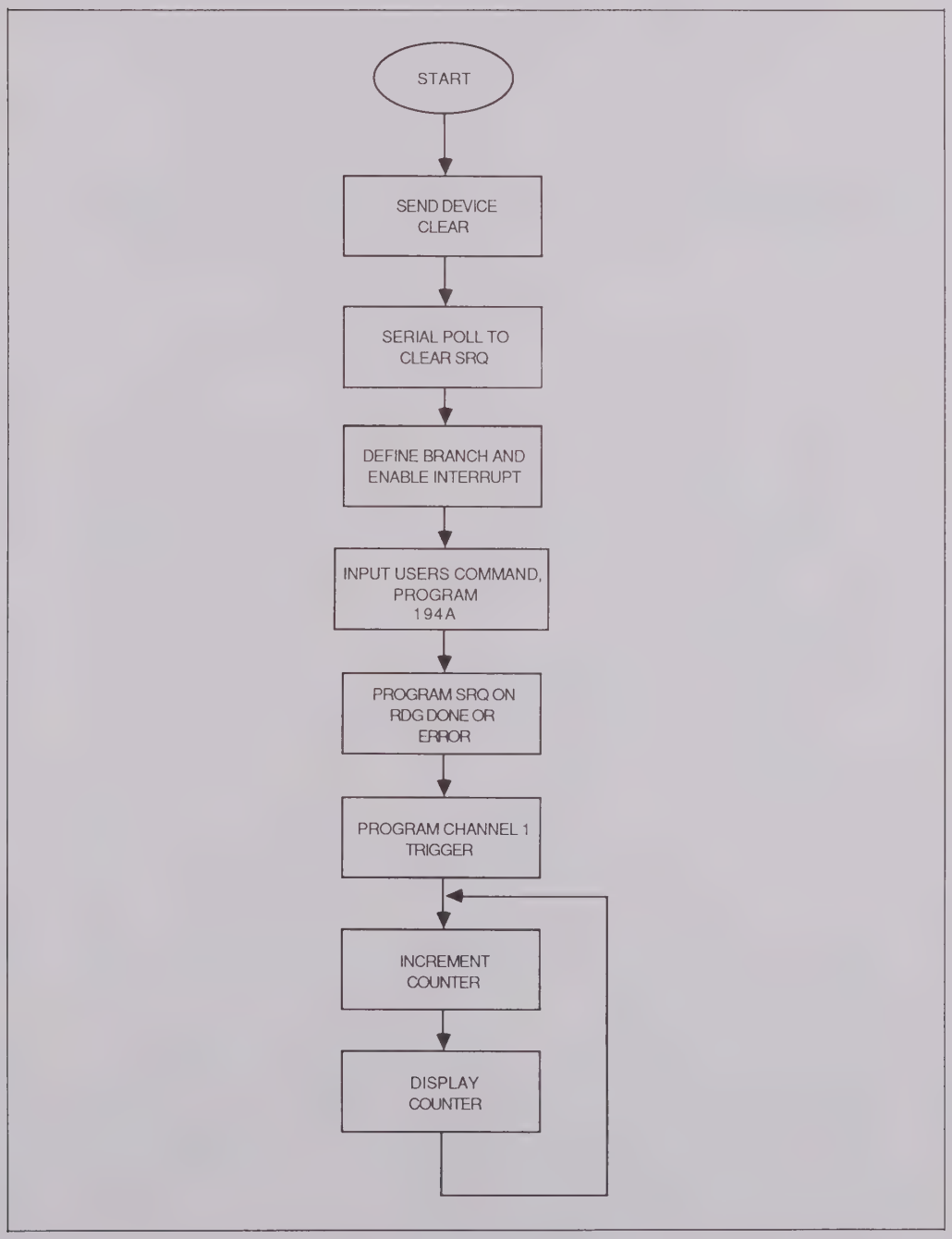


Figure 9. Program 4 Flowchart (Initialization and Main Task Section)

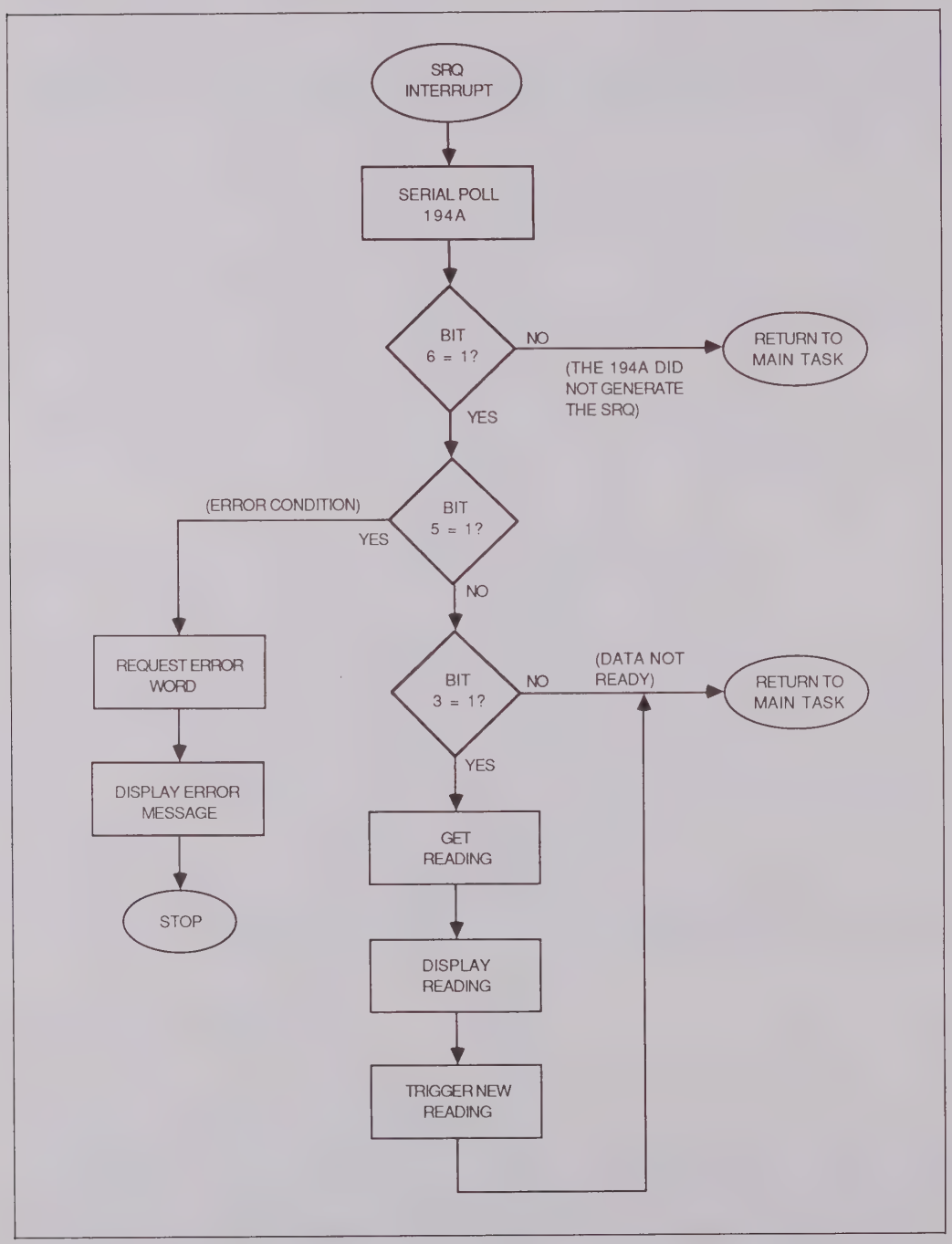


Figure 10. Program 4 Flowchart (SRQ and Error Handling Section)

USING POLLED SRQ

The program can be modified for polled SRQ by making the modifications below:

1. Remove lines 70, 80, 90, 100, and 550.
2. Add the following lines to the program:

```
211 STATUS 7;7;Bus_lines                ! Read bus lines.
212 IF BIT(Bus_lines)=1                 ! Check SRQ bit status.
    THEN GOSUB Service_routine
```

With this modification, the controller must check the status of the SRQ line each time it goes through its main task loop. Although from a user's standpoint the program may appear to operate the same, the additional controller overhead necessary for polled SRQ may become restrictive in larger applications.

Program 4. SRQ and Error Handling

```
10 Program_setup:
20      CLEAR 709                               ! Return the 194A to a known state
30      Serial_pol=SPOLL(709)                   ! Clear out any pending SRQs
40      DIM Error_status$[50]                   ! Dimension Error_status string
50      DIM Message$[50]                        ! Dimension Message string
60      DIM Reading$[50]                        ! Dimension Reading string
70      Hpib=7                                  ! Define interface select code
80      Mask=2                                  ! Define bit mask
90      ON INTR Hpib GOSUB Service_routine      ! Define interrupt initiated branch
100     ENABLE INTR Hpib;Mask                   ! Enable interrupt on SRQ
110 Model_194_setup:!
111     REMOTE 709
120     OUTPUT 709;"C2T7X"          ! Disable ch2 by setting for external trigger
130     PRINT "ENTER COMMAND"       ! Prompt for command to be entered
140     INPUT Command$              ! Input Command string
150     OUTPUT 709;"C1X"&Command$&"X"     ! Output users command
160     OUTPUT 709;"M40X"           ! Program SRQ for error and reading done.
170     OUTPUT 709;"T2X"            ! Program for trigger on GET and arm A/D.
180     TRIGGER 709                 ! Trigger a reading
190 Main_task:  !
200     N=N+1                       ! Users main task goes here
210     PRINT N                     ! Users main task goes here
220     GOTO Main_task              ! Users main task goes here
230     STOP
240 Service_routine:!
250     Serial_pol=SPOLL(709)       ! Perform serial poll on the 194A.
260     PRINT " SERIAL POLL = ";Serial_pol              ! Print serial poll byte
270     IF BIT(Serial_pol,6)=0 THEN RETURN              ! Determine if the 194A
280                                                     !  generated the SRQ.
290     IF BIT(Serial_pol,5)=1 THEN GOSUB Error         ! Determine if an error
300                                                     !  condition occurred.
310     IF BIT(Serial_pol,3)=1 THEN GOSUB Take_reading ! Determine if a
320                                                     !  reading is done.
330     RETURN
340 Error:!
350     OUTPUT 709;"U1X"            ! Select the 194A error conditions.
360     ENTER 709;Error_status$     ! Read the 194A error conditions.
370                                 ! Lines 380 to 490 determine type of error.
380     IF Error_status$[4,4]="1" THEN Message$="IDDC ERROR"
390     IF Error_status$[5,5]="1" THEN Message$="IDDCO ERROR"
```

Program 4. SRQ and Error Handling (Cont.)

```
400     IF Error_status$[6,6]="1" THEN Message$="NO REMOTE"
410     IF Error_status$[7,7]="1" THEN Message$="CH1 TRIGGER OVERRUN"
420     IF Error_status$[8,8]="1" THEN Message$="CH2 TRIGGER OVERRUN"
430     IF Error_status$[10,10]="1" THEN Message$="FAILED SELFTEST"
440     IF Error_status$[12,12]="1" THEN Message$="CHANNEL 2 NOT INSTALLED"
450     IF Error_status$[13,13]="1" THEN Message$="WAVEFORM DATA INVALID"
460     IF Error_status$[14,14]="1" THEN Message$="NO CHANGES ALLOWED"
470     IF Error_status$[15,15]="1" THEN Message$="TRANSLATOR ERROR"
480     IF Error_status$[16,16]="1" THEN Message$="SAMPLES CONFLICT"
490     IF Error_status$[17,17]="1" THEN Message$="DELAY CONFLICT"
500     PRINT Message$              ! Print error message.
510     STOP                        ! Stop program if error occurred.
520 Take_reading:!
530     ENTER 709;Reading$          ! Take a reading
540     PRINT Reading$              ! Print reading
550     ENABLE INTR 7               ! Enable interrupts
560     TRIGGER 709                 ! Trigger another reading
570     RETURN                      ! Return to main task
580     END
```

BINARY DATA TRANSFER

PROGRAMMING CONCEPTS

- Data transfer speed considerations.
- Transferring binary data from the Model 194A to the computer.
- Converting binary data into displayable form.

PROGRAMMING CONSIDERATIONS

When a relatively large number of samples must be transferred from the Model 194A to the computer, speed becomes an important consideration. The binary transfer mode of the instrument can significantly reduce the amount of time it takes to send a given amount of data from the instrument to the computer. Binary transfer speeds up this operation for two reasons: (1) the Model 194A is not required to process the data before sending it out, and (2) fewer bytes are transferred per sample in the binary mode.

In order to take full advantage of binary transfer speed, the computer must be capable of transferring at a rate at least as high as that of the Model 194A--approximately 90K bytes per second in binary mode. The HP9816 used for the example program below can operate at rates as high as 130K bytes per second using fast handshake transfer, but other controllers may require use of DMA transfer to operate with sufficient speed. In any event, the advantage to using binary transfer can be seen by comparing the time necessary to transfer 1000 samples. In binary mode using fast handshake, these samples can be transmitted in only 65msec as opposed to 15 seconds required when using ASCII transfer and the ENTER statement. (using an HP9816 computer).

The general binary formats for the G6 and G7 modes are shown in Figure 11. In addition to the one or two bytes for each sample, a 4-byte status word and byte count (G7 only) is added to the front of each byte sequence. The status word contains important information such as range, channel, function, as well as the exponent for the data. In order to display the data, it must first be converted into ASCII form, as demonstrated in Program 5 below.

PROGRAM OPERATION

Program 5 below demonstrates fundamental concepts for obtaining binary data and converting it into displayable form. Figure 12 shows a general flowchart of the program.

Program highlights include:

1. The program first initializes variables and I/O paths (lines 10-70).
2. The instrument is then programmed for the waveform mode (F0), number of samples (N0), sampling rate (S0), binary data format (G7), and reading buffer disabled (lines 80-140).

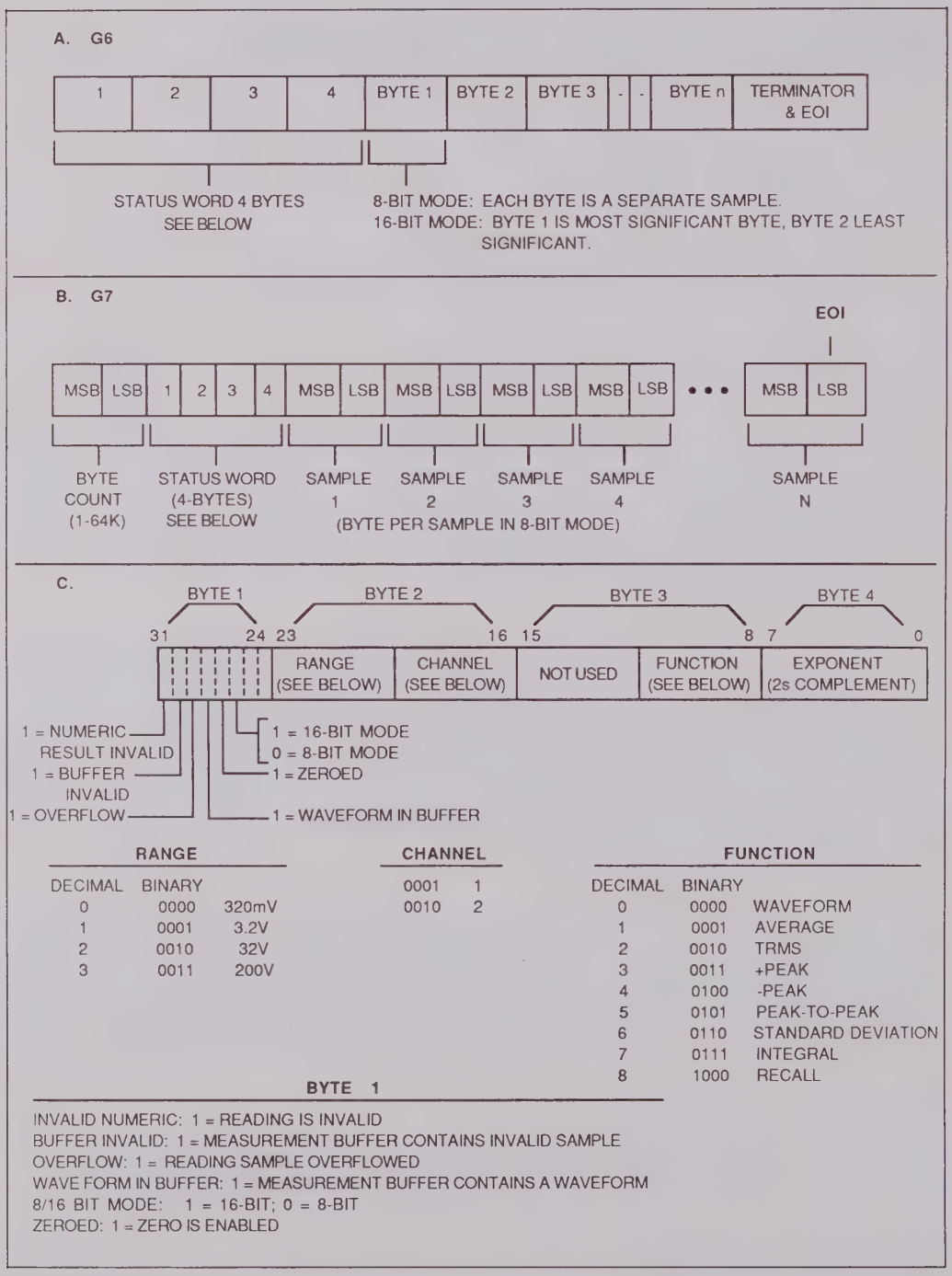


Figure 11. Binary Data Formats

3. Next, the program performs a test to see if the reading is done (lines 160 and 170). This test is performed by serial polling the unit and testing the state of bit 3 in the status byte (the reading done bit). If the bit is cleared, the reading (and thus the measurement) has not been completed, and the program remains in the loop until bit 3 is set, indicating that a reading is available.
4. Once it has been determined that the reading is complete, the fast handshake transfer is performed to read the binary data into a computer I/O buffer (lines 180-221). In order to properly terminate the transfer, it is necessary to know how many bytes are to be transferred. This value is determined from the number of samples programmed, as well as whether the data is in eight or 16-bit format. A value of six is added to the total byte count to account for the count and status bytes.
5. Next, the byte count and status are read from the I/O buffer and placed into numeric variables (line 240). In this case, the first two bytes represent the byte count, while the last four bytes contain Model 194A status information, as shown in Figure 11(c).
6. Binary data is then read from the I/O buffer into a numeric array. (line 270 or 280). The transfer is in byte (8-bit) or word (16-bit) format, depending on the sampling rate (the instrument stores 8-bit data at rates above 100kHz). Note that the end of the data for this transfer is determined by EOI.
7. The binary data is then converted into displayable form (lines 350-410). This subroutine must take into account the weight of each bit. In the 8-bit mode, the bit weight is 256, while in the 16-bit mode, the bit weight is 1.
8. The data is then scaled according to range by using the range information located in byte 2 of the status word. Since this range information is located in the most significant nibble, it must be converted by ANDing with 48 and then dividing by 16. Figure 13 outlines this conversion process.
9. Once final test is made to determine the status of the overflow bit located in byte 1 (lines 520-570). A normal or overflow reading message is determined by the status of this bit.

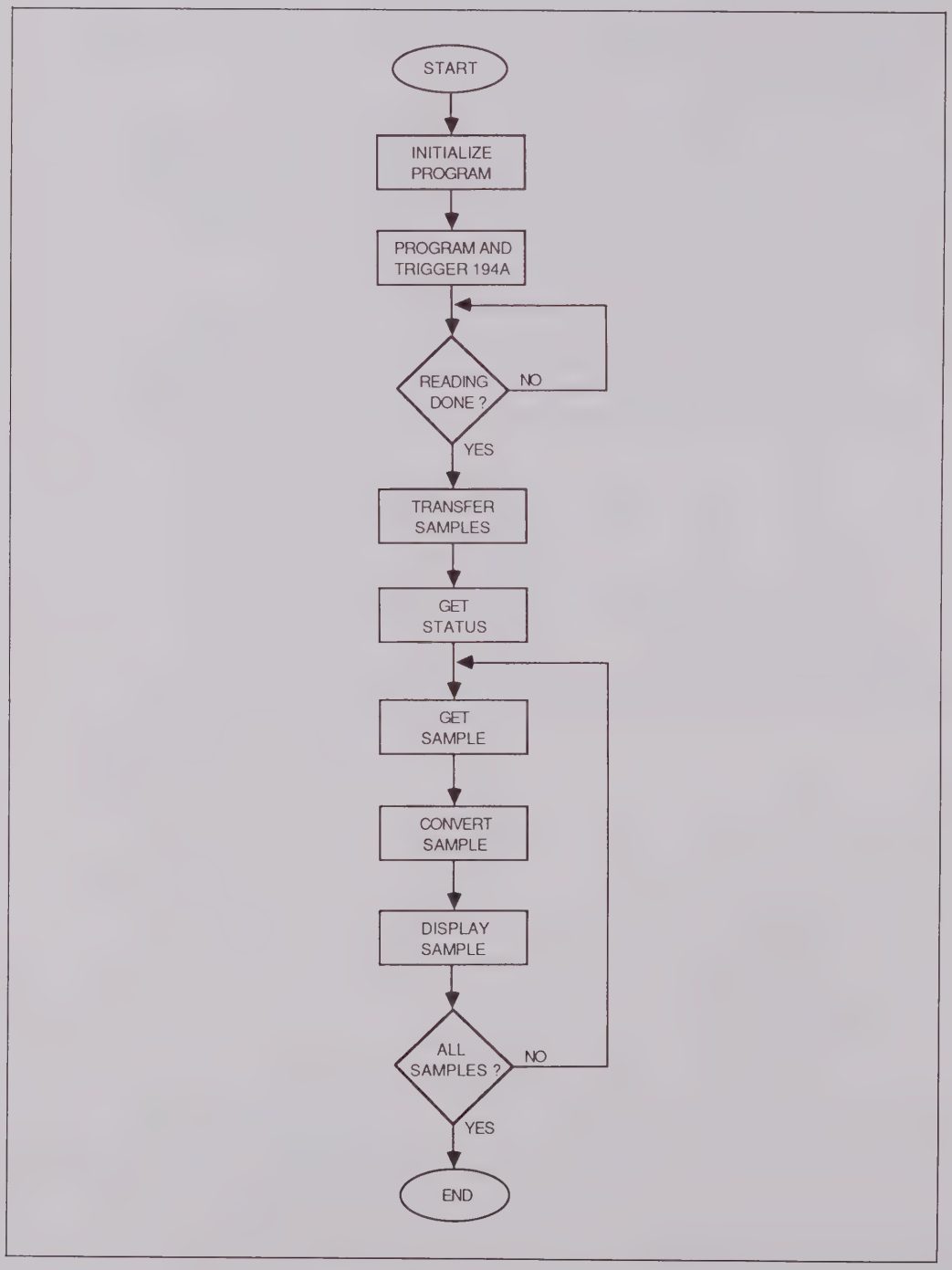


Figure 12. Flowchart of Binary Transfer Program

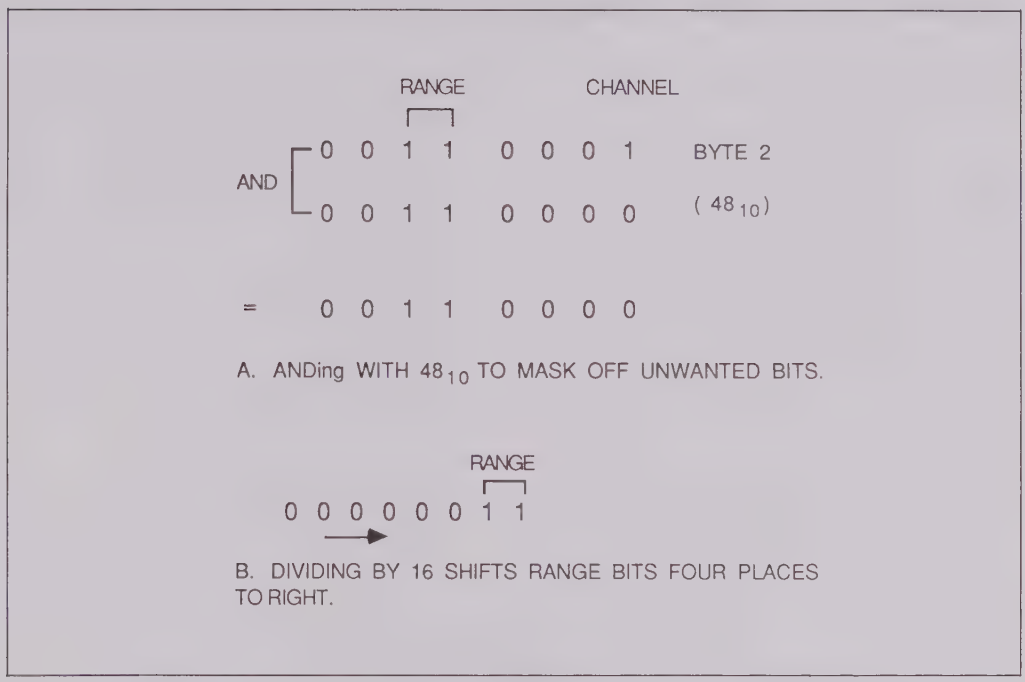

RANGE
CHANNEL
AND
0 0 1 1 0 0 0 1
BYTE 2
0 0 1 1 0 0 0 0
(48 10)
= 0 0 1 1 0 0 0 0
A. ANDing WITH 48 10 TO MASK OFF UNWANTED BITS.
RANGE
0 0 0 0 0 0 1 1
B. DIVIDING BY 16 SHIFTS RANGE BITS FOUR PLACES TO RIGHT.

Figure 13. Converting Byte 2 Range Bits

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Execute the program by pressing the RUN key.
3. Apply a DC voltage of less than 320mV.
4. When the prompt for number of samples is displayed, type in the desired value. For example, for 100 samples type in 100 and press the ENTER key.
5. When the computer prompts for the reading rate, type in the desired value. For example, to program a 1μsec rate interval, type in 1E-6 and press ENTER.
6. The Model 194A will now process a measurement and then display the individual samples, including sample number and a message as to whether the sample is normal or overflowed. With less than 320mV applied, the reading should always be normal.
7. Increase the applied voltage to 500mV and run the program again. Note that the overflow message is now displayed because the 500mV applied signal is above the maximum input voltage of the 320mV range.
8. Change the R1 range command in line 130 of the program to R0. This change will program the Model 194A for autorange. Re-run the program several times with different applied voltages between 200mV and 200V and note that displayed data is automatically scaled according to range.

Program 5. Binary Data Transfer

```
10 Initialize_prog: !
30       INTEGER B(32000)                          ! Dimension integer
40       REAL A                    ! Reserves storage for floating point variable
41       DIM A$[32000] BUFFER                      ! Dimension buffer
50       ASSIGN @D194 TO 709                       ! Assign I/O path name to 194A
60       ASSIGN @Buf TO BUFFER A$                  ! Assign I/O path name to BUFFER
70       OUTPUT KBD;CHR$(255)&CHR$(75);            ! Clear screen
80 Initialize_194: !
85       REMOTE 709                                ! Put 194A in remote
90       PRINT "ENTER NUMBER OF SAMPLES"           ! Prompt for # samples
100      INPUT Samples                             ! Input # samples
110      PRINT "ENTER READING RATE IN uS (e.g 1E-6)"  ! Prompt for rate
120      INPUT Rate                                ! Input rate
121                ! Program 194
130      OUTPUT 709;"F0R1N0,"&VAL$(Samples)&"S0,"&VAL$(Rate)&"G7Q0X"
140      OUTPUT 709;"T27X"                         ! Arm the 194A
150 Test_reading:!
160      S=SPOLL(709)                              ! Perform serial poll
170      IF BIT(S,3)=0 THEN GOTO 160               ! Test for reading done
180 Setup_transfer: !
190      IF Rate<1.0E-5 THEN Number_bytes=1        ! Determine # bytes per sample
200      IF Rate>1.0E-5 THEN Number_bytes=2        ! Determine # bytes per sample
210 Perform_transf: !
211                ! Transfer 194A Data using fast handshake. Tell how many
212                !   bytes are to be transfered.
220      TRANSFER @D194 TO @Buf;COUNT (Number_bytes*Samples)+6,WAIT
221                ! Read 194A status from buffer
230 Read_status:!
240      ENTER @Buf USING "%,W,B,B,B,B";Count,Status1,Status2,Status3,Status4
250 Read_data:!
260      FOR Data=1 TO Samples
270      IF Rate<1.0E-5 THEN ENTER @Buf USING "%,B";B(Data) ! Read 8 bit word
271                                                         !   from buffer.
280      IF Rate>1.0E-5 THEN ENTER @Buf USING "%,W";B(Data) ! Read 16 bit word
281                                                         !   from buffer.
290      GOSUB Convert              ! Gosub to routine that converts raw binary
300      PRINT Data,Reading,Message$
310      NEXT Data
320      SEND 7;UNT
330      STOP
340 Convert:!
350      A=B(Data)
360      IF Rate<1.0E-5 THEN Bit_weight=256          ! Determine bit weight for 8
361                                                  !   bit mode.
370      IF Rate>1.0E-5 THEN Bit_weight=1            ! Determine bit weight for 16
371                                                  !   bit mode.
380      IF A>0 THEN R=(A*Bit_weight)-32768          ! Calculate real number.
390      IF A<0 THEN R=(A*Bit_weight)+32768          ! Calculate real number .
400      GOSUB Scale
410      RETURN
420 Scale:!
430      Mask=48                                     ! Define mask byte
440      D=BINAND(Status2,Mask)                      ! AND Status2 with Mask
450      Range=D/16                                  ! Determine value of range
451                                                  !   status
460      IF Range=0 THEN Reading=R*1.0E-5            ! Scale reading for.320V range
470      IF Range=1 THEN Reading=R*1.0E-4            ! Scale reading for 3.2V range
480      IF Range=2 THEN Reading=R*1.0E-3            ! Scale reading for 32V range
490      IF Range=3 THEN Reading=R*1.0E-2            ! Scale reading for 200V range
```

Program 5. Binary Data Transfer (Cont.)

```
500       GOSUB Oflo                                  ! Goto to Oflo test routine
510       RETURN
520 Oflo:!
530       Mask=32                                     ! Define mask byte
540       J=BINAND(Status1,Mask)                      ! AND Status1 with mask
550       IF J=32 THEN Message$="OVERFLOW READING"    ! If overflow bit is set give
551                                                   !   overflow message
560       IF J<>32 THEN Message$="NORMAL READING"     ! If overflow bit is not set
561                                                   !  give normal reading message
570       RETURN
580    END
```

READING BUFFER OPERATION

PROGRAMMING CONCEPTS

- Fundamental operation of the reading buffer.
- Disabling the unused channel so that only desired readings are stored in the reading buffer.
- Triggering each stored reading individually.
- Operation of the reading buffer pointer.
- Dumping entire buffer contents in one string.

PROGRAMMING CONSIDERATIONS

The reading buffer is a 100 location buffer into which processed readings can be stored as they become available. By sending the appropriate command, you can operate the reading buffer in a linear configuration (Q1) or a circular mode (Q2). With a linear buffer, the instrument will continue to store readings until all 100 locations are full and then stop storing until the buffer is completely emptied. With the circular buffer, the instrument will store 100 readings and then stop. As old readings are taken from the buffer (in oldest to newest order), new readings will be stored in the empty locations.

Since a single 100-location reading buffer is shared by both channels (dual-channel units), care must be taken when reading the buffer to determine the source of the data. If both channels are running, data will be stored in the buffer on an as-available basis. Thus, to determine which channel is the data source, you should check the channel suffix (Figure 2) on the reading string. Of course, you can also turn off an unused channel by programming for a trigger condition that will not be met. For example, to turn off channel 2, send C2XT7X over the bus. Doing so will make certain that data comes only from the desired channel.

The way the buffer fills will depend on the programmed trigger mode. In the immediate mode, the instrument will fill the buffer without requiring additional triggers. However, in the remaining trigger modes, you must supply one trigger per measurement in order to fill the buffer.

Once your data is stored in the buffer, there are several ways to access it. With the G0 through G2 data formats, one buffer location will be sent with each request. However, you can also dump the complete buffer in one operation by using G4 or G5. Remember, however, that the computer input buffer or string must be dimensioned large enough to handle all the bytes in the data string.

By using the B3 command, you can dictate how many of the reading buffer locations will be sent. Simply use B3,nX, where n represents the last location that will be transmitted. For example, if you send the command B3,20X, locations 1 through 20 will be transmitted when requested.

It is important to clarify the operation of the buffer location suffix, which is shown in Figure 2. This value does not represent the buffer location number; rather, it shows the number of readings left in the reading buffer after accessing this particular location. For example, assume you send B3,20X to set an end pointer of 20. The first location to be sent will be number 1, but the suffix will show 0019, meaning that there are still 19 locations left to access. This value will decrement once per buffer location until it shows a value of 0000 with the access of the last buffer location.

PROGRAM OPERATION

Program 6 below demonstrates fundamental operation of the reading buffer. Figure 14 is a flowchart of the program.

Basically, the program operates as follows:

1. The string called Reading$ is dimensioned to hold 2,000 bytes (line 10), and the unit is placed in remote (line 20).
2. Both channel 1 and channel 2 are disarmed by sending T7X to the appropriate channel (lines 30 and 40). Channel 1 is disarmed to ensure that no readings are being processed when the reading buffer is turned on; otherwise, undesired readings will be stored. Channel 2 is turned off to avoid storing channel 2 readings at all.
3. Channel 1 is programmed to take 10 samples (N0,10) per measurement at the maximum rate of 1μsec (S0,1E-6) by line 50.
4. The reading buffer is then enabled for the linear configuration by sending Q1X (line 70).
5. Next, a 100-point loop is set up to arm the A/D converter and trigger each measurement (lines 80-120). Each time through the loop, the A/D converter is armed by sending T3X, and a measurement is then triggered by GET. As each reading is processed, it is stored in the reading buffer in the next sequential location.
6. After the loop is complete, the unit is programmed to dump the first 20 readings of the reading buffer in one long string (line 130). The G5 command tells the instrument to output all available buffer locations at once, while the B3,20 command limits the output to the first 20 locations only (locations 1-20).
7. Finally, the reading string is requested and displayed on the computer screen (lines 160-170).

RUNNING THE PROGRAM

1. Enter the program into the computer.
2. Press the RUN key to execute the program.
3. The program will send commands to the Model 194A, and 100 readings will be stored in the reading buffer.
4. Once storage is complete, the complete reading string will be displayed on the computer display. Note that the reading number suffixes decrement from 19 down to 0 throughout the reading string. As pointed out earlier, this number represents the number of readings left to access at that point--not the buffer location number.

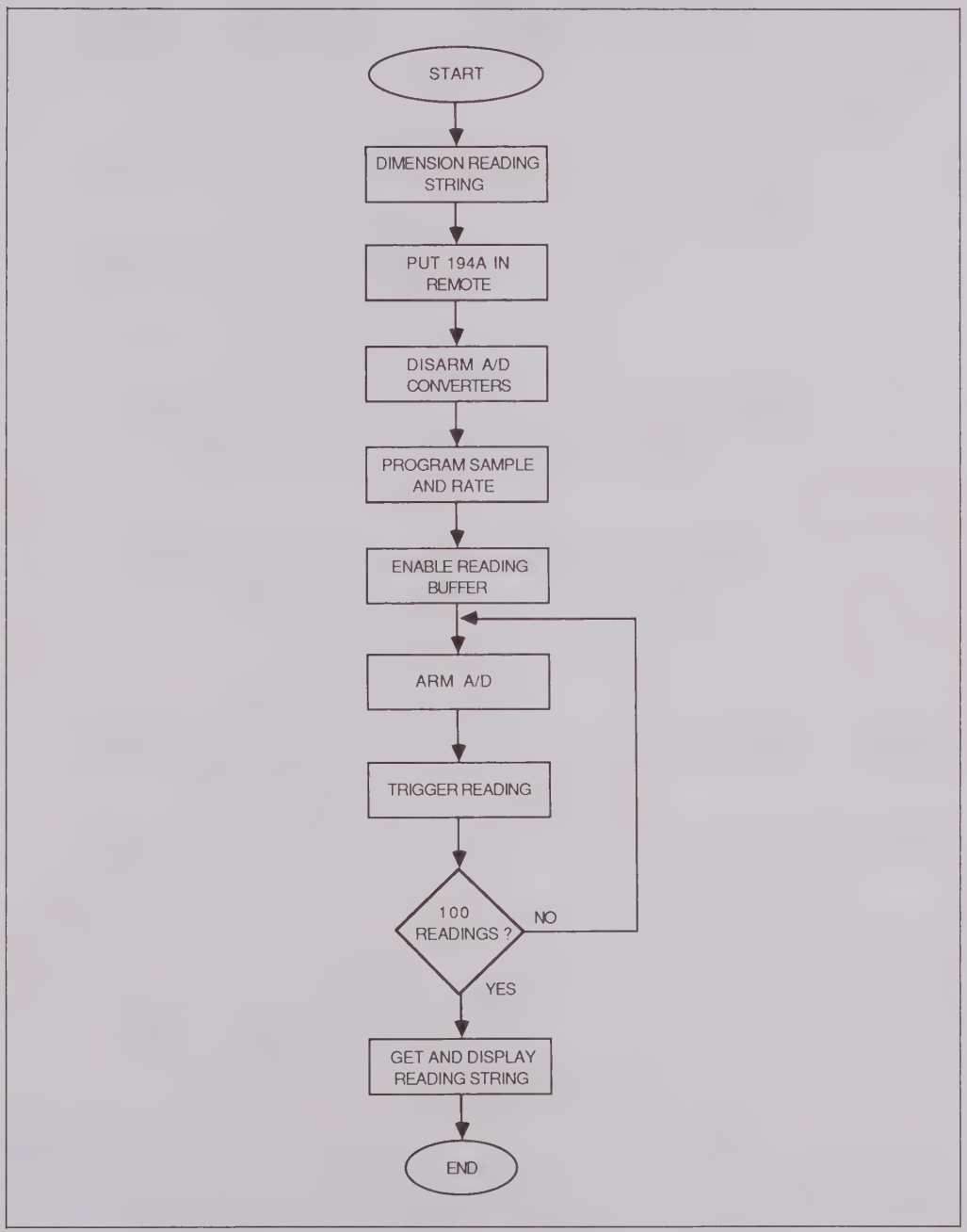


Figure 14. Flowchart for Reading Buffer Program

Program 6. Reading Buffer Operation

```
10    DIM Reading$[2000]                    ! Dimension reading$
20    REMOTE 709                            ! Put the 194A in remote
30    OUTPUT 709;"C1T7X"                    ! Turn off channel 1
40    OUTPUT 709;"C2T7X"                    ! Turn off channel 2
50    OUTPUT 709;"C1N0,10S0,1E-6X"          ! Program rate and # samples for
60                                          !   channel 1.
70    OUTPUT 709;"Q1X"                      ! Enable linear buffer
80    FOR I=1 TO 100                        ! Take 100 readings
90    OUTPUT 709;"T3X"                      ! Program trigger mode and arm A/D
100                                         !   channel 1.
110   TRIGGER 709                           ! Trigger the 194A
120   NEXT I                                ! Trigger another reading
130   OUTPUT 709;"G5B3,20X"                 ! Program 194A to send
140                                         !   readings 1-20
150                                         !
160   ENTER 709;Reading$                    ! Read buffer locations 1-20
170   PRINT Reading$                        ! Print reading$
180   END
```

USING TRANSLATOR

PROGRAMMING CONCEPTS

- Using Translator to make programs easier to read.
- Instrument emulation using Translator.

PROGRAMMING CONSIDERATIONS

The Translator feature allows you to substitute a group of ASCII characters for a device-dependent command string. Basically, there are two chief reasons for using this feature: (1) to make programs more readable, and (2) to emulate the operation of other instruments.

Translator words are defined with the ALIAS command by including the defined word along with the associated device-dependent command string. For example, to define the word SAMPLES to program 1000 samples, the following would be used:

```
OUTPUT 709;''ALIAS SAMPLES N0,1000X ;''
```

Note that spaces are included to delimit the ALIAS command, the defined word (SAMPLES), and the device-dependent command (N0,1000X). Also, the definition string must be terminated with a semicolon.

Once Translator words are defined, using them is simply a matter of sending them in place of device-dependent commands, for example:

```
OUTPUT 709;''SAMPLES''
```

In this case, the word SAMPLES would program 1000 samples because SAMPLES was previously defined as N0,1000X.

MAKING PROGRAMS READABLE

Program 7 below gives an example of how Translator can be used to make programs more readable. Figure 15 is a flowchart of the program.

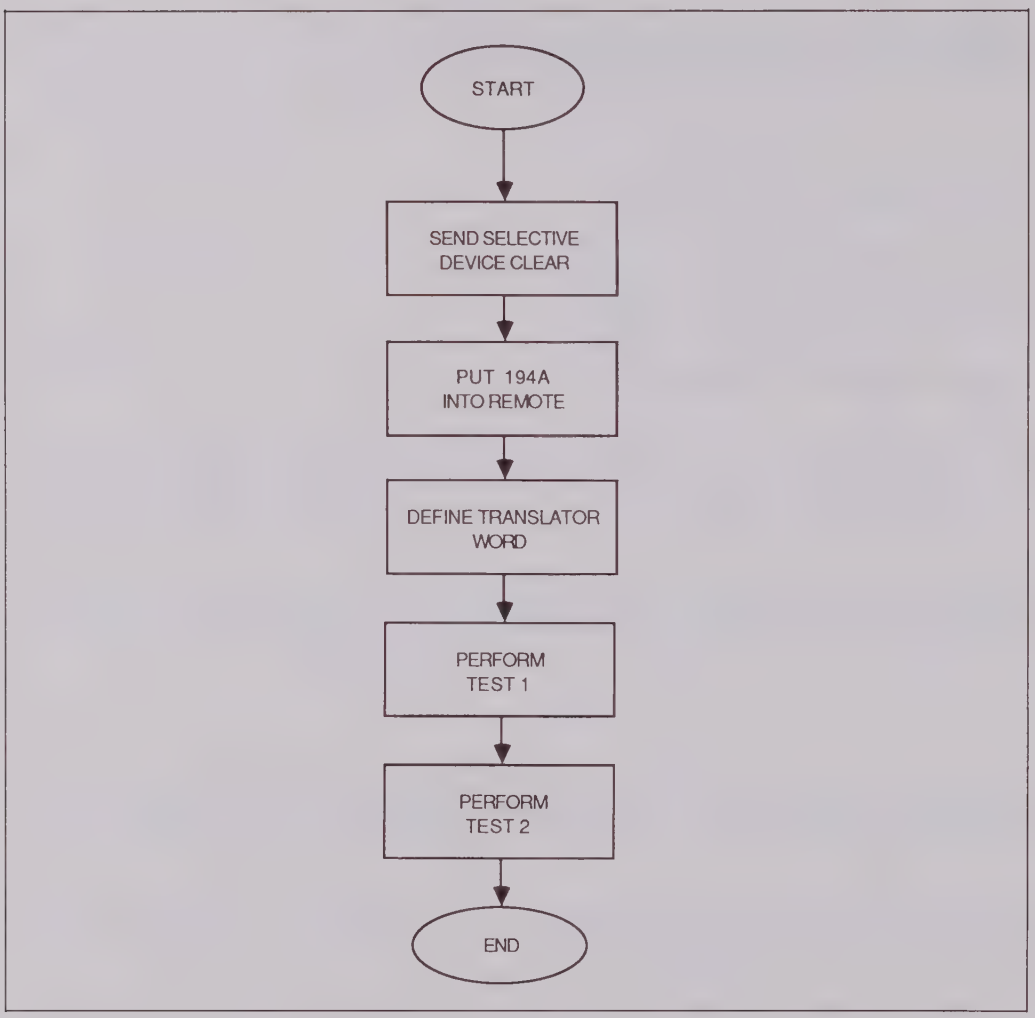


Figure 15. Flowchart for Program 7

Program Operation

Key points of the program include:

1. The variable Voltmeter is defined as 709 for use in subsequent OUTPUT statements (line 20).
2. The instrument is then returned to default conditions and placed in remote (lines 30 and 40).
3. Next, the Translator words are defined by using the ALIAS command (lines 50-100).
4. Test 1 is then executed by sending the appropriate Translator words to program the various operating modes (lines 120-170).
5. In a similar manner, Test 2 is performed by programming the instrument (lines 190-240).

Running the Program

1. Enter the program into the computer.
2. Press RUN to execute the program.
3. The program will then send Translator words to the instrument to be defined. Two different readings should appear on the computer CRT, one each for Test 1 and Test 2.

INSTRUMENT EMULATION

Program 8 demonstrates the use of Translator to emulate the TRMS measurement of an AC waveform. Note that TCR4? is the command string that is to be emulated by the Model 194A. This command selects RMS, 10VAC range, and DC coupling of the DVM being emulated. Figure 16 gives a flowchart of the program.

Program Operation

1. The Translator word is first defined (line 3).
2. The instrument is placed in remote (line 10).
3. The defined Translator word (TCR4?) is then sent to the instrument (line 20).
4. Finally, a loop is set up to request and display 10 readings (lines 40-70).

Running the Program

1. Enter the program into the computer.
2. Apply an AC signal of approximately 10V RMS to the instrument INPUT jack.
3. Press the RUN key to execute the program.
4. The program will send the necessary Translator definition command to the instrument, program the instrument accordingly, and then request and display 10 readings.

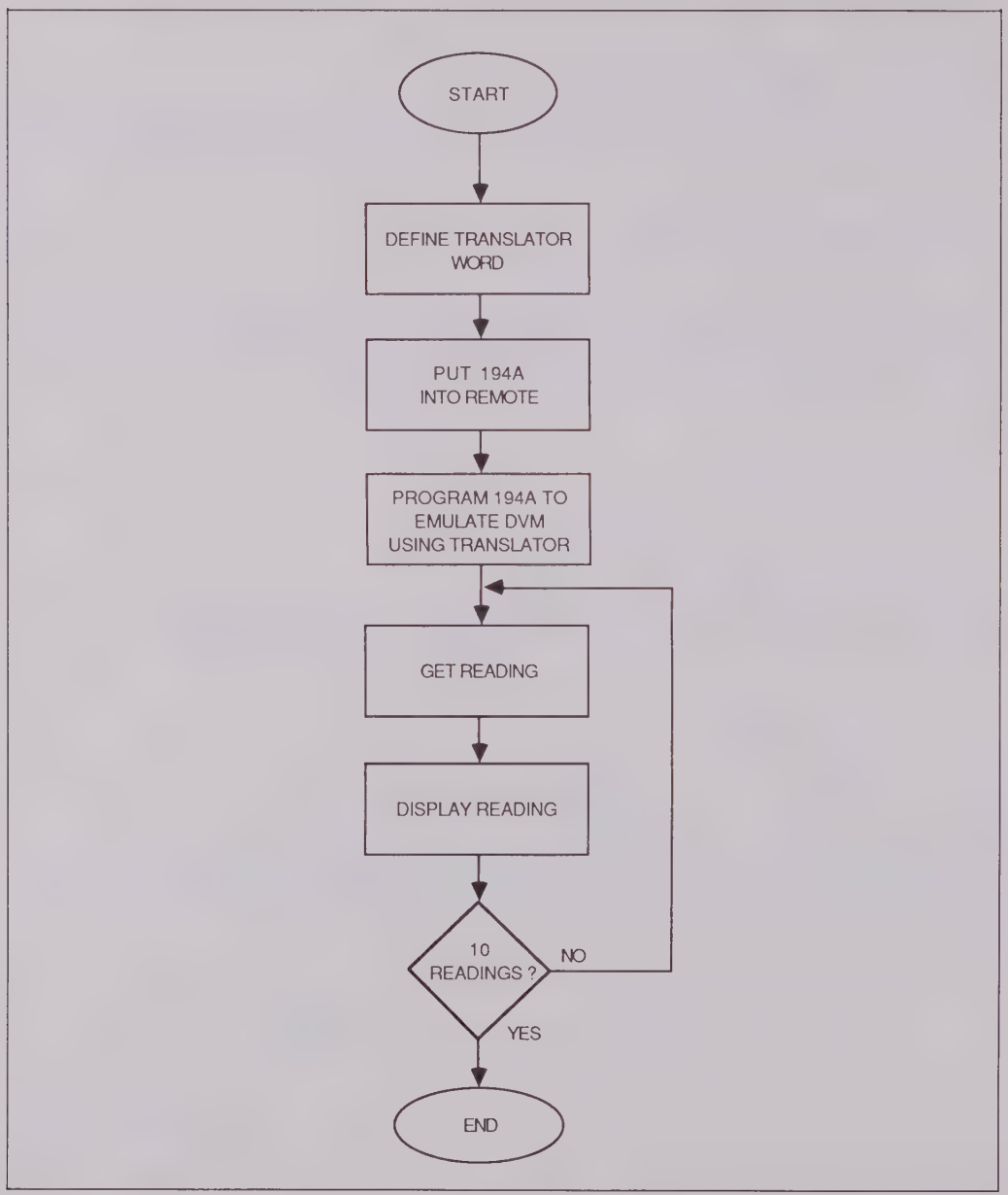


Figure 16. Program 8 Flowchart

Program 7. Using Translator for Program Readability

```
10 Program_setup:
20    Voltmeter=709                                        ! Define 194A address
30    CLEAR Voltmeter                                      ! Return to default
40    REMOTE Voltmeter                                     ! Put 194A in remote
50    OUTPUT Voltmeter;"ALIAS ZERO C1Z4X ;"                ! Define translator word
60    OUTPUT Voltmeter;"ALIAS SETUP1 C1XF3R3I1X ;"         ! Define translator word
70    OUTPUT Voltmeter;"ALIAS SETUP2 C1XF1XR2X ;"          ! Define translator word
80    OUTPUT Voltmeter;"ALIAS 1000SAMPLES N0,1000X ;"      ! Define translator word
90    OUTPUT Voltmeter;"ALIAS ARM T26X ;"                  ! Define translator word
100   OUTPUT Voltmeter;"ALIAS DISABLECH2 C2XT7X ;"         ! Define translator word
110 Test_1:!
120   OUTPUT Voltmeter;"DISABLECH2"                        ! Disable channel 2
130   OUTPUT Voltmeter;"SETUP1"                            ! Program setup 1
140   OUTPUT Voltmeter;"ZERO"                              ! Zero reading
150   OUTPUT Voltmeter;"ARM"                               ! Arm the A/D
160   ENTER Voltmeter;Reading$                             ! Take reading
170   PRINT Reading$                                       ! Print reading
180 Test_2:  !
190   OUTPUT Voltmeter;"SETUP2"                            ! Program setup 2
200   OUTPUT Voltmeter;"1000SAMPLES"                       ! Program 1000 samples
210   OUTPUT Voltmeter;"ARM"                               ! Arm the A/D
220   ENTER Voltmeter;Reading$                             ! Take reading
230   PRINT Reading$                                       ! Print reading
240   END
```

Program 8. Using Translator for Instrument Emulation

```
1     ! Line 3 translates the DVM commands to 194A commands.
3     OUTPUT 709;"ALIAS TCR4? C2T7XC1F2R3G1I0S0,166.7E-6N0,1000T26X ;"
10    REMOTE 709                                  ! Put DVM in remote
20    OUTPUT 709;"TCR4?"                          ! Program DVM for continuous mode,
30                                                !   RMS AC ,DC coupled and 10V range.
40    FOR I=1 TO 10                               ! Loop 10 times
50    ENTER 709;Reading$                          ! Get reading
60    PRINT Reading$                              ! Print reading$
70    NEXT I
80    END
```

USING EXTERNAL SAMPLE TRIGGERING

PROGRAMMING CONCEPTS

- Using external sample triggering to test potentiometers.
- Fundamental graphing techniques.
- Combining two instruments on the bus.

PROGRAMMING CONSIDERATIONS

The external sample trigger feature of the Model 194A allows you to synchronize sample collection to external events. By applying appropriate TTL-compatible pulses to the sample trigger input, you can trigger each sample individually.

One possible external sample trigger application involves testing for potentiometer linearity using the basic setup shown in Figure 17. In addition to the Model 194A, additional equipment is necessary: (1) a Model 220 current source, and (2) an optical encoder. The Model 220 sources current through the potentiometer, developing a voltage to be measured by the Model 194A. The optical encoder develops the sample trigger pulses necessary to trigger individual samples.

To run the test, the current source is set to the desired value, and the potentiometer shaft is rotated, either manually, or by a motor. As the shaft rotates, the encoder develops sample trigger pulses at precise increments of rotation (typically 1 to 2 degrees per trigger pulse). At each rotational increment, the Model 194A measures the voltage across the potentiometer and stores that value as a single sample.

Once all samples are taken, it is a simple matter to compute the resistance at each point from the measured voltage and the source current. Computed data can then be further analyzed or graphed, as required. Program 9 below demonstrates basic techniques for graphing such data.

PROGRAM OPERATION

Program 9 is written to perform the potentiometer test described above. Figure 18 shows a flowchart outlining key sections of the program.

Important program sections are described below.

1. The program is first initialized by clearing the screen and assigning I/O paths (lines 20-70).
2. Next, prompts are given for the user to input degree increments, degrees of span, and maximum potentiometer resistance (lines 80-130). From this information, the total number of samples required is calculated based on the degrees of span and degree increments (line 140).

3. The Model 220 current and voltage compliance are then programmed, and the output is turned on (lines 170 and 180). In this instance, we are programming a current of 100μA (1.00E-4); to program a different current value, simply change the current variable in line 170 to the required value.
4. The Model 194A is programmed for the number of samples (based on the required number computed), sampling rate (166.7μsec), range (32V), function (waveform), data format (no prefix) by lines 230-260. In this case, we are using channel 2 with channel 1 disabled. To use channel 1 instead of channel 2, reverse the C1 and C2 commands in these three lines.
5. Now that the instrument is ready, the program prompts the operator to turn the potentiometer and then inputs all samples into an array called Reading (lines 310-360).
6. The plot routines are then initialized (lines 380-530).
7. The graph is labelled (lines 550-740).
8. Finally, the resistance is calculated from the stored samples, and the deviation from nominal is then plotted (lines 760-860).

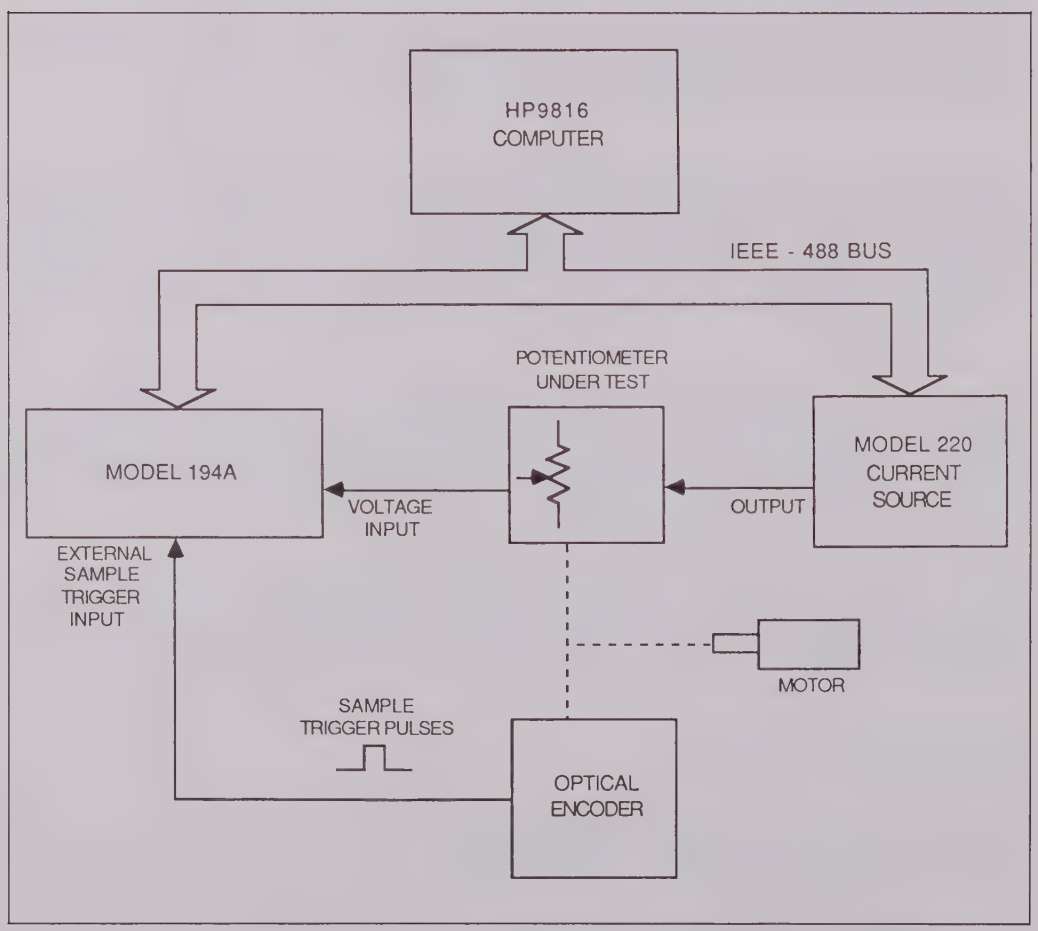


Figure 17. Potentiometer Test Configuration (Set up for Program 9)

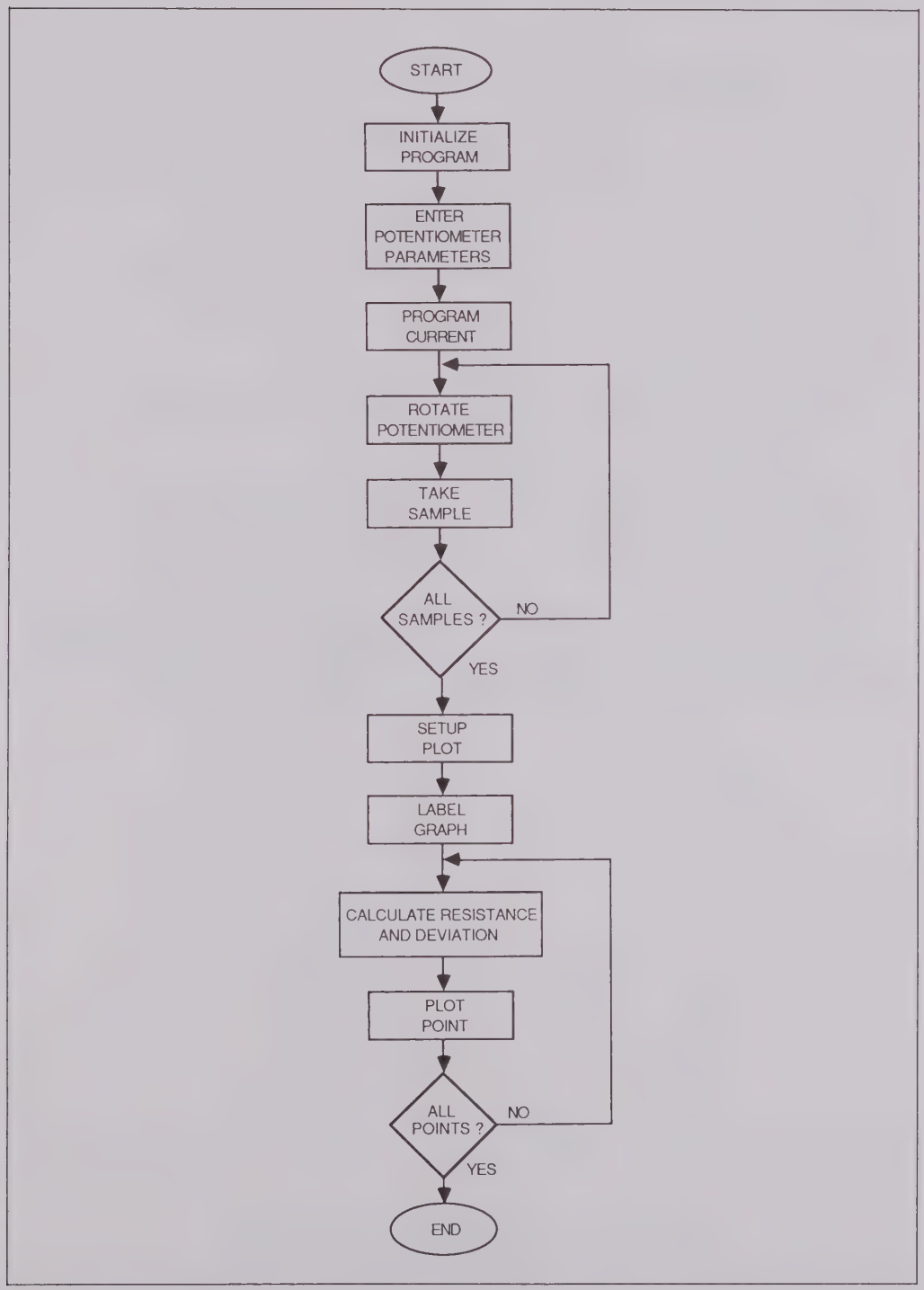


Figure 18. Potentiometer Test Program Flowchart

RUNNING THE PROGRAM

1. Connect the Model 220 OUTPUT and Model 194A VOLTAGE INPUT to the potentiometer being tested. Both lows should be connected to one end of the potentiometer, and both highs should be connected to the wiper.
2. Mount the optical encoder on the potentiometer shaft, and connect the encoder output to the SAMPLE TRIGGER input of the Model 194A through the Model 1942 or similar sample cable. Remember that the encoder must supply TTL-compatible pulses with a duration of greater than 100nsec (samples trigger on the rising edge).
3. Connect the instruments to the IEEE-488 bus of the controller using suitable cables.
4. Be sure that the primary address of the Model 194A is set to 9, and that the Model 220 address is programmed for a value of 12.
5. Enter the program into the computer, and check it for errors. If a different current value than 100μA is desired, modify line 170 accordingly.
6. Rotate the potentiometer to one end of its rotation.
7. Make sure the ABORT line on the Model 194A is set low (connect the black and white wires on the Model 1942 Sample Cable together).
8. Execute the program by pressing the RUN key.
9. At the prompts, enter the degree increments (the resolution in degrees of the encoder), the total span of the pot in degrees, and the maximum potentiometer value.
10. When prompted to do so, smoothly rotate the potentiometer to the opposite end of rotation. The instrument will collect samples as the potentiometer is rotated.
11. Once data is taken, the readings will be read into the computer array, and the graph will be generated using the voltage data and current source value to compute the resistance. The plot generated shows deviation from nominal (percent) versus degrees of rotation, as shown in the example of Figure 19. Note that a 20-turn potentiometer was used for this example, yielding a total possible rotation in excess of 6,000 degrees.

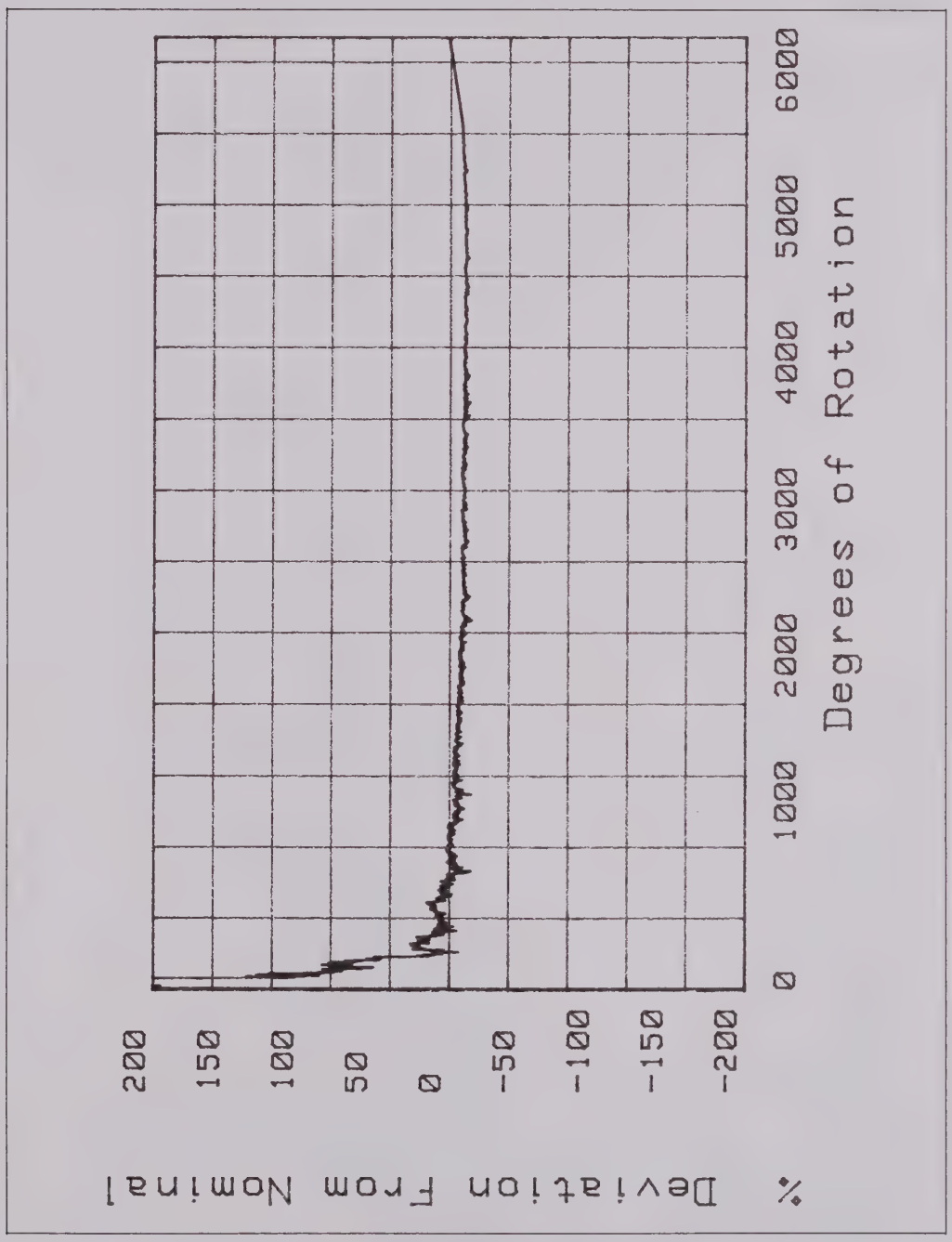


Figure 19. Potentiometer Test Example

Program 9. Potentiometer Test

```
10 Initalize_prgm:!
20     GCLEAR                              ! Clear graphics display
30     OUTPUT KBD;CHR$(255)&CHR$(75)       ! Clear screen
40     ASSIGN @M220 TO 712                 ! Assign I/O path name to 220
50     ASSIGN @M194 TO 709                 ! Assign I/O path name to 194A
60     CLEAR @M194                         ! Return 194A to a known state
70     DIM Reading(10000)                  ! Dimension reading
80     PRINT "ENTER THE NUMBER OF DEGREES BETWEEN EACH READING"
90     INPUT Degree                        ! Input degree
100    PRINT "ENTER THE TOTAL NUMBER OF DEGREES OF SPAN"
110    INPUT Degree_total                  ! Input Degree_total
120    PRINT "ENTER MAX RESISTANCE OF POT IN OHMS"
130    INPUT Resistance                    ! Input resistance
140    Samples=Degree_total/Degree         ! Calculate number of samples
150    OUTPUT KBD;CHR$(255)&CHR$(75)       ! Clear screen
160 Set_up_220:!
170    Current=1.00E-4                     ! Define test current value
180    REMOTE @M220                        ! Put 220 in remote mode
190    OUTPUT @M220;"I"&VAL$(Current)&"V60F1X" ! Program 220 for voltage,
200                                        !    current and operate mode.
210 Setup_194a: !
220    REMOTE @M194                        ! Put 194A in remote mode
230    OUTPUT @M194;"C1T7X"                ! Turn off channel 1
240    OUTPUT @M194;"C2S0,1.667E-4N0,"&VAL$(Samples)&"X"! Program channel 2 for
250                                                     !    rate and samples
260    OUTPUT @M194;"C2X6IR3F0B0T27X"      ! Program channel 2 for prefix and
270                                        !    suffix off,32V range, waveform
280                                        !    mode, clear waveform output and
290                                        !    single arming continuous trigger.
300 Buffer_fill:!
310    PRINT "TURN POT"                    ! Prompt to start turning pot
320    FOR Count=1 TO Samples              ! Define number of readings
330                                        ! Controller will wait for buffer to
340                                        !   fill
350    ENTER 709;Reading(Count)            ! Get 194A reading  and put into array
360    NEXT Count                          ! Repeat for next reading
370 Setup_plot:!
380    GRAPHICS ON                         ! Turn graphics display on
390    OUTPUT KBD;CHR$(255)&CHR$(75)       ! Clear screen
400    GINIT                               ! Set graphic variables to default
410    PLOTTER IS 702,"HPGL"               ! Select plotting device
420    CSIZE 7                             ! Set character size
430    MOVE 5,90                           ! Move pen
440    LABEL " Model 194A Potentiometer Test"   ! Draw label
450    CSIZE 4                             ! Set character size
460    MOVE 40,10                          ! Move pen
470    LABEL "Degrees of Rotation"         ! Draw label
480    MOVE 5,15                           ! Move pen
490    DEG                                 ! Select degrees as unit of measure
500    LDIR 90                             ! Define angle at which label is drawn.
510    LABEL "%  Deviation From Nominal"      ! Draw label
520    VIEWPORT 20,100,20,70               ! Define area
530    GRID 10,10,0,50,1,1,1               ! Draw grid
540 Label_axis:  !
550    VIEWPORT 20,100,20,70               ! Define area
560    WINDOW 0,Degree_total,-200,200      ! Define current unit of measure
570    CLIP OFF                            ! Disable soft clip limit
580    CSIZE 3                             ! Set character size
590    FOR Label_x=0 TO Degree_total STEP 1000    ! Label X axis at 1000 degree
```

Program 9. Potentiometer Test (Cont.)

```
600                                                    !   increments
610     DEG                                  ! Set degrees as unit of measure
620     LDIR 0                               ! Set angle at which label is drawn
630     MOVE Label_x,-240                    ! Move pen
640     LABEL USING "#,K";Label_x            ! Draw label
650     NEXT Label_x                         ! Repeat for next label
660     FOR Label_y=-200 TO 200 STEP 50      ! Label Y axis at 50% increments
670     DEG                                  ! Set degrees as unit of measure
680     LDIR 360                             ! Set angle at which label is drawn
690     MOVE -700,Label_y                    ! Move pen
700     LABEL USING "#,K";Label_y            ! Draw label
710     NEXT Label_y                         ! Repeat for next label
720     VIEWPORT 20,100,20,70                ! Define area
730     WINDOW 1,Samples,-200,200            ! Define current unit of measure
740     FRAME                                ! Draw frame
750 Calculate:!
760     FOR Count=1 TO Samples*Degree        ! Define reading array size
770     IF Reading(Count)=0 THEN Reading(Count)=.00001 ! Prevent undefined math
780                                                    !    error condition
790     Ohms=Reading(Count)/Current                    ! Calculate actual
800                                                    !    resistance
810     Nominal=(Count/Samples)*Resistance             ! Calculate ideal
820                                                    !    resistance
830     Error=(Nominal-Ohms)/Ohms*100                  ! Calculate % of error
840 Plot:!
850     PLOT Count,Error                               ! Plot point
860     NEXT Count                                     ! Repeat for next point
870   END
```

GENERATING BUS TIME INFORMATION

PROGRAMMING CONCEPTS

- Factors that affect trigger to SRQ times.
- Demonstrating measurement of trigger to SRQ times.
- Summarizing typical GET trigger to SRQ times.

PROGRAMMING CONSIDERATIONS

The time it takes for a reading to be completed after the instrument is triggered is an important consideration, especially in time-critical situations. A common yardstick used to measure this time period is the duration between the trigger stimulus and a reading done SRQ generated by the instrument, as shown in Figure 20. Note that the duration between the trigger is the sum of the measurement time and the reading processing time, as we will now discuss.

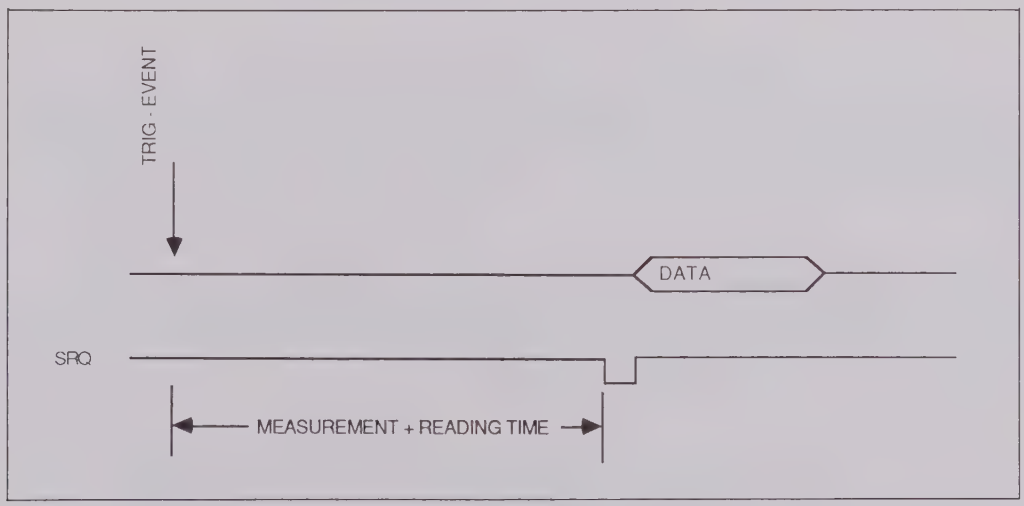


Figure 20. Basic Time Interval Between Trigger and SRQ

The measurement time depends solely on the programmed number of samples and the sampling rate. For example, with a sampling interval of 1μsec and 1000 sample, the measurement time is 1msec (1μsec X 1000 = 1msec). The reading processing time is primarily due to software overhead, and it depends on the selected math function as well as the number of samples in the measurement. Other time periods such as trigger response times do enter into the equation, but they are usually small enough to be negligible, and can be ignored for all practical purposes.

A common operational condition is to trigger the Model 194A with GET (Group Execute Trigger), and then wait until the instrument generates an SRQ, as shown in Figure 21. (For simpler situations, the controller can simply wait until the SRQ occurs, but more complex situations require interrupt SRQ processing.) Here, the time interval is defined as the period from when the GET trigger is completed until the time that SRQ is set low by the instrument when the reading is completed.

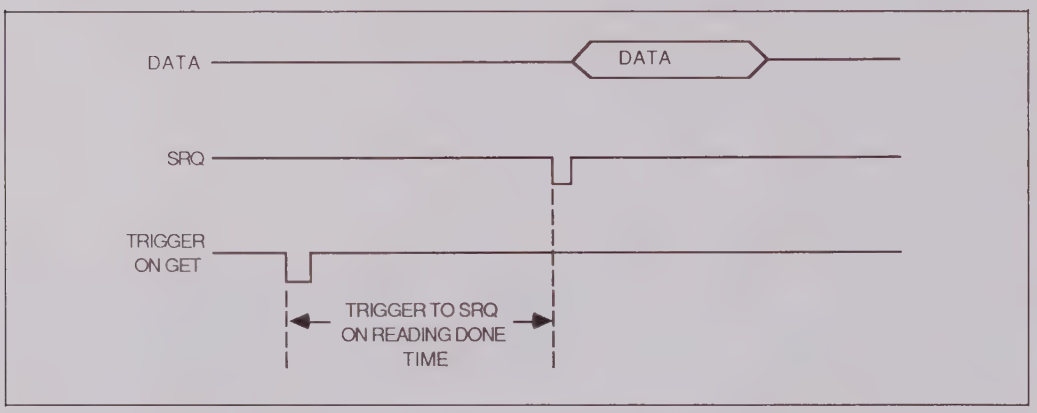


Figure 21. Simplified GET-to-SRQ Interval

In order to accurately measure GET-to-SRQ times, it is necessary to monitor the ATN, DAV, and SRQ lines with a logic analyzer. Figure 22 shows typical waveforms for triggering the unit with GET and waiting for a reading done SRQ. Note that the presence of GET is detected by monitoring the DAV line, as activity on this line indicates that bytes are being sent over the data lines (one byte per spike). In this case, three spikes are evident: one each for the listen address, GET command byte, and UNL command.

Once the instrument receives the GET, it will initiate a measurement, take all samples in that measurement, process the reading, and then generate the SRQ. Typically, the instrument will also be serial polled (to verify the reason for the SRQ), and this activity is also shown on the ATN and DAV lines in Figure 22.

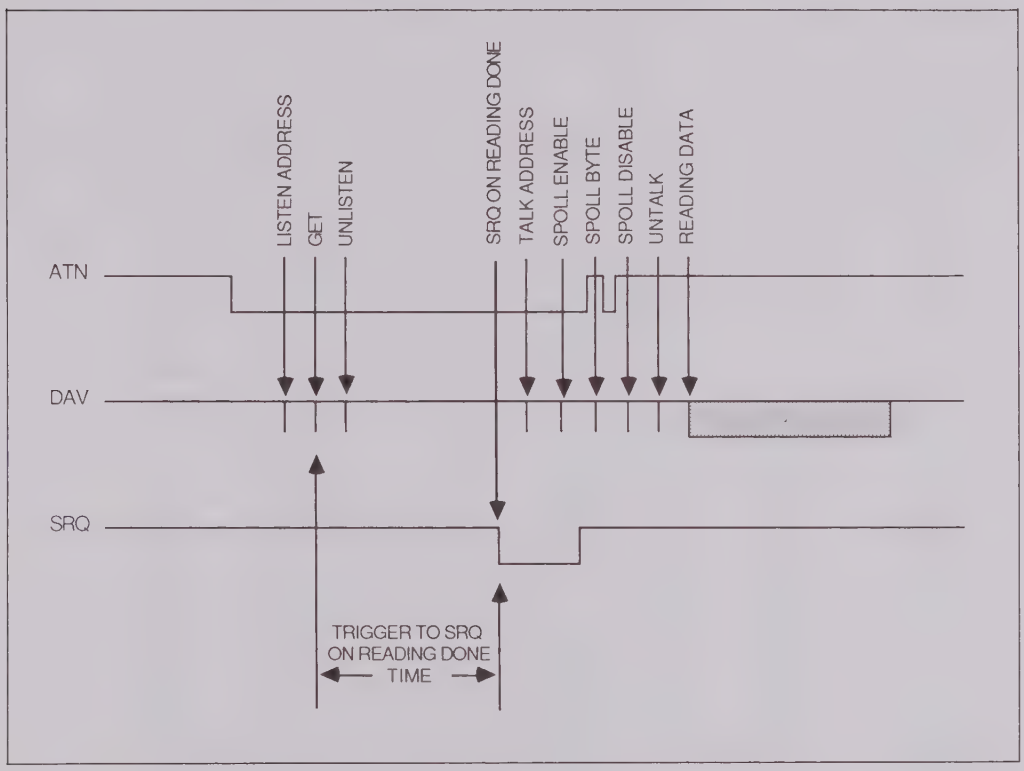


Figure 22. GET-to-SRQ Times Showing Serial Poll and Data Transfer

PROGRAM OPERATION

Program 10 is a typical program that can be used to generate GET to reading done SRQ times. Figure 23 is a flowchart of the program. Typical times for various sampling rates, measurement durations, and math functions are summarized in Table 3.

Program 10 operates as follows:

1. The program first dimensions variables, assigns I/O paths, and places the instrument in remote (lines 20-50).
2. Next, the Model 194A is serial polled in order to clear out any pending SRQ (line 60).
3. The user is then prompted to enter the desired command, and the instrument is then programmed accordingly (lines 80-111). Note that channel 2 is disabled, and channel 1 is programmed for the 320mV range (R1), SRQ on reading done (M8), and the continuous, GET mode (T2).
4. The instrument is then triggered by sending GET (line 130), and the program enters a loop to wait for the SRQ to occur (lines 150 and 160).
5. Once the SRQ is detected, the Model 194A is serial polled (line 170), and a data string is requested and displayed along with the serial poll byte (lines 190 and 200).
6. The program then loops back (line 210) to repeat the trigger, SRQ, serial, data display process. The loop repeats indefinitely to allow the bus lines to be analyzed properly.

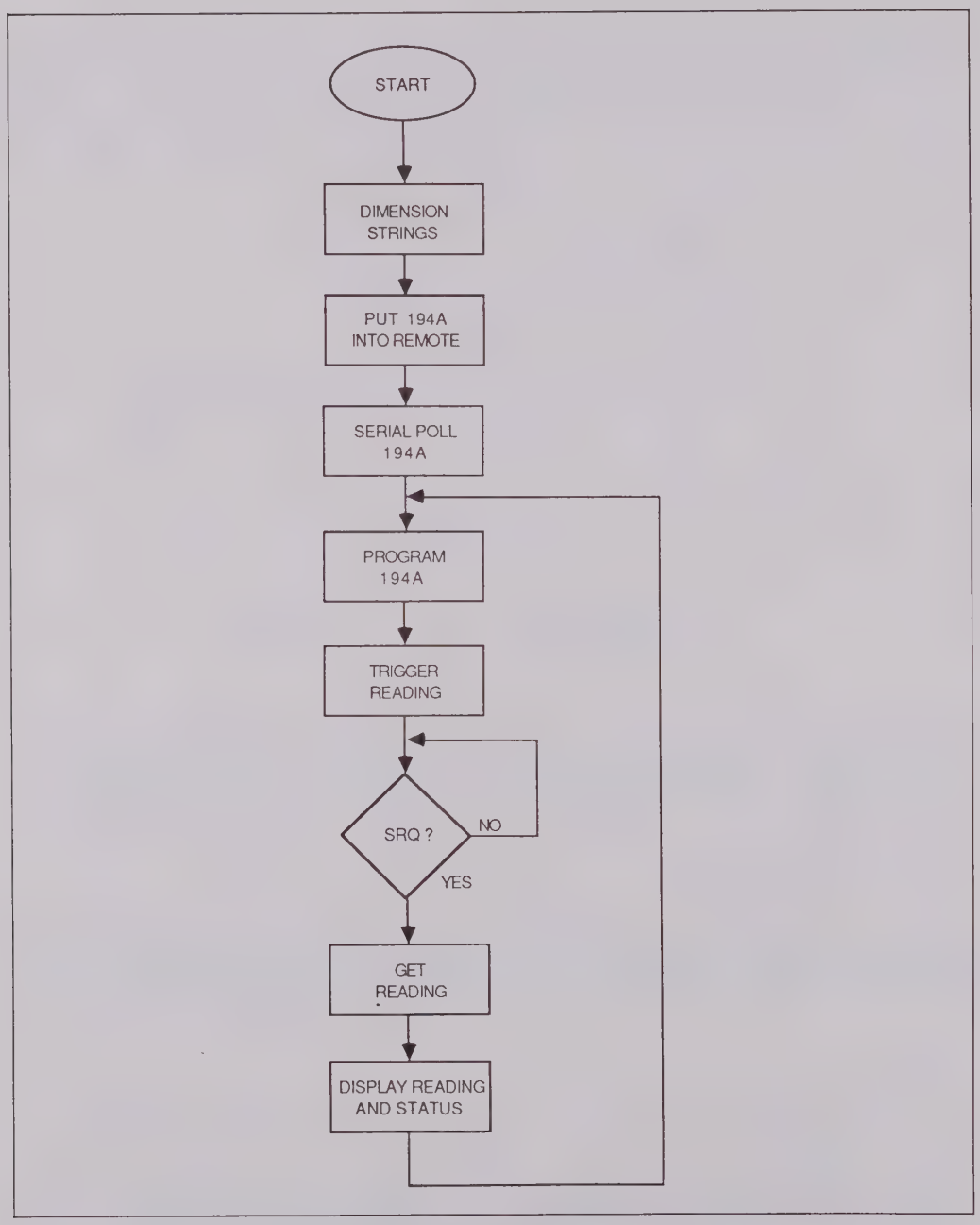


Figure 23. Program 10 Flowchart

Table 3. Typical GET to SRQ on Reading Done Times

Sample Interval	# of Samples	Math Function	Trigger to SRQ on Reading Done	Sample Interval	# of Samples	Math Function	Trigger to SRQ on Reading Done
1μs	1000	AVERAGE	136ms	10μs	100	AVERAGE	24ms
1μs	1000	TRms	255ms	10μs	100	TRms	143ms
1μs	1000	+PEAK	59.5ms	10μs	100	+PEAK	15ms
1μs	1000	−PEAK	59.5ms	10μs	100	−PEAK	15ms
1μs	1000	P-P	59.5ms	10μs	100	P-P	15ms
1μs	1000	STD DEV	337ms	10μs	100	STD DEV	153ms
1μs	1000	INT	156ms	10μs	100	INT	44.4ms
1μs	100	AVERAGE	23ms	100μs	1000	AVERAGE	248ms
1μs	100	TRms	145ms	100μs	1000	TRms	347ms
1μs	100	+PEAK	14ms	100μs	1000	+PEAK	154ms
1μs	100	−PEAK	14ms	100μs	1000	−PEAK	154ms
1μs	100	P-P	14ms	100μs	1000	P-P	154ms
1μs	100	STD DEV	153ms	100μs	1000	STD DEV	425ms
1μs	100	INT	44ms	100μs	1000	INT	249ms
10μs	1000	AVERAGE	139ms	100μs	100	AVERAGE	32.8ms
10μs	1000	TRms	258ms	100μs	100	TRms	151.5ms
10μs	1000	+PEAK	65ms	100μs	100	+PEAK	23.8ms
10μs	1000	−PEAK	65ms	100μs	100	−PEAK	238ms
10μs	1000	P-P	65ms	100μs	100	P-P	23.8ms
10μs	1000	STD DEV	335ms	100μs	100	STD DEV	161.5ms
10μs	1000	INT	157ms	100μs	100	INT	53.4ms

RUNNING THE PROGRAM

1. Connect a logic analyzer to the ATN, DAV, and SRQ lines, and set up the analyzer to view bus information.
2. Enter the program into the computer.
3. Execute the program by pressing the RUN key.
4. At the prompt, enter the desired sample rate, number of samples, and math function in command form. For example, to program a rate of 1μsec, 1000 samples, and the average math function enter the following: "S0,1E-6N0,1000,F1" Note that it is not necessary to include the X character as the program will add it to the string when it is sent to the instrument (you must use quotes, however because of the commas).
5. A reading will then be triggered, and the program will wait for the SRQ on reading done to occur. The instrument will be serial polled and asked to send a reading, after which both will be displayed on the computer CRT. The loop will be repeated indefinitely, allowing you to use the logic analyzer to measure trigger to SRQ times.

Program 10. Test Program for Measuring Trigger to SRQ on Reading Done

```
10 System_init:    !
20    DIM Command$[50]                          ! Dimension Command$
30    DIM Data$[100]                            ! Dimension Data$
40    ASSIGN @M194 TO 709                       ! Assign I/O path name to 194A
50    REMOTE @M194                              ! Put 194A into remote
60    Ser_poll=SPOLL(@M194)                     ! Serial poll 194A to clear out
61                                              !    old SRQ.
70 Program_inst:    !
80    INPUT "ENTER COMMAND :",Command$          ! Enter command$
110   OUTPUT @M194;"C1"&Command$&"R1M8T2X"      ! Output command$ to 194A
111   OUTPUT @M194;"C2T7X"                      ! Disable channel 2
120 Trig_inst:  !
130   TRIGGER @M194                             ! Trigger the 194A
140 Wait_for_srq:   !
150   STATUS 7,7;6piblines                      ! Return interface status
160   IF BIT(6piblines,10)=0 THEN 150           ! Test SRQ bit
170   Ser_poll=SPOLL(@M194)                     ! Serial poll the 194A
180 Read_inst:   !
190   ENTER @M194;Data$                         ! Take reading from 194A
200   PRINT Data$,Ser_poll                      ! Print data and serial poll
201                                             !    byte.
210   GOTO 130                                  ! Trigger another reading
220   END
```

APPENDIX A
Device-Dependent Commands

FUNCTION	F0	Waveform
	F1	Average
	F2	TRMS
	F3	+Peak
	F4	-Peak
	F5	Peak-to-peak
	F6	Standard deviation
	F7	Integral
	F20	CH1-CH2
	F21	CH1/CH2
RANGE	R0	Auto
	R1	320mV
	R2	3.2V
	R3	32V
	R4	200V
	R12	Cancel auto (stay on present range)
RATE	S0,m	m=sampling rate in sec. (1E-6≤m≤1sec)
	S1,m	m=sampling rate in Hz (1≤m≤1E6Hz)
NUMBER SAMPLES	N0,m	m=number of samples (1≤m≤65,535, 8-bit, 1≤m≤32,767, 16-bit)
	N1,m	m=measurement duration in sec (0≤m≤65.534E3, 8-bit; 0≤m≤32,766, 16-bit)
TRIGGER	T0	Continuous, talk
	T1	Single, talk
	T2	Continuous, GET
	T3	Single, GET
	T4	Continuous, X
	T5	Single, X
	T6	Continuous, external
	T7	Single, external
	T20,m	Continuous, +slope, level m (−200≤m≤+200V)
	T21,m	Single, +slope, level m (−200≤m≤+200V)
	T22,m	Continuous, -slope, level m (−200≤m≤+200V)
	T23,m	Single, -slope, level m (−200≤m≤+200V)
	T24	Continuous, other channel
	T25	Single, other channel
	T26	Continuous, immediate
	T27	Single, immediate
	T30	Start plotter
	T31	Stop plotter
DELAY	W0±m	Delay in number of samples (65,536≤m≤1E7)
	W1,±m	Delay in sec. (65,536≤m≤1E7)

NOTE: The following characters may be used as a delimiter in place of comma: ! @ # $ % ^ & ( ) = \ / < > ? : ; <space>

Device-Dependent Command Summary (Cont.)

DATA FORMAT	G0	ASCII, 1 rdg, prefix on, suffix off
	G1	ASCII, 1 rdg, prefix off, suffix off
	G2	ASCII, 1 rdg, prefix on, suffix on
	G3	ASCII, n rdgs, prefix on, suffix off
	G4	ASCII, n rdgs, prefix off, suffix off
	G5	ASCII, n rdgs, prefix on, suffix on
	G6	Binary, n rdgs, prefix off, suffix off
	G7	Binary, n rdgs (byte count), prefix off, suffix off
ANALOG OUTPUT	O0	Abort plotter
	O1	Test Analog Output (all outputs=full scale)
	O2,0	XY Mode off
	O2,1	CRT Mode
	O2,2	Scope Mode
	O2,3	Future Expansion
	O2,4	Slow Plot (XY Analog Plotter)
	O2,5	Strip chart
	O3,0	Measurement Buffer
	O3,1	64k RAM Buffer
	O3,2	Display Readings
	O3,3	IEEE-488 Reading Buffer
	O4,m	m=X Output Full Scale Voltage ($1 \leq m \leq 10$)
	O5,m	m=Y Output Full Scale Voltage ($1 \leq m \leq 10$)
	O6,m	m=Z Output Blanking Level (1=High, 0=Low)
	O7,m	m=XY Zoom Value ($0.1 \leq m \leq 1000$)
	O8,m	m=XY Pan Value ($-65{,}536 \leq m \leq 65{,}536$)
	O9	Start Plotter
BUFFER POINTER	B0	Clear waveform output, decrement reading buffer pointer.
	B1,m	Set measurement buffer start pointer to m.
	B2,m	Set measurement buffer end pointer to m.
	B3,m	Set reading buffer start pointer to m.
READING BUFFER	Q0	Reading buffer off.
	Q1	Linear reading buffer on.
	Q2	Circular reading buffer on.
FILTER	P0	Filter off
	P1	500kHz filter
	P2	50kHz filter
ZERO	Z0	Disable zero
	Z1	Enable zero
	Z2	Use next measurement as baseline
	Z3,m	Use value m as baseline value ($-200 \leq m \leq +200V$)
	Z4	Z1 + Z2
	Z5,m	Z1 + Z3 ($-200 \leq m \leq +200V$)
INPUT COUPLING	I0	DC coupling
	I1	AC coupling
	I2	Ground coupling

Device-Dependent Command Summary (Cont.)

STATUS	U0	Current channel status
	U1	Error status
	U2	Data status
	U3	Zero (Z) parameter
	U4	Delay (W) parameter
	U5	Number samples (N) parameter
	U6	Sample rate (S) parameter
	U8	Trigger level
	U9	Measurement buffer start pointer
	U10	Measurement buffer end pointer
	U11	Reading buffer start pointer
	U12	Get Translator list
SRQ	M0	SRQ disabled
	M1	Overflow
	M2	Data condition
	M4	Button pressed
	M8	Reading done
	M16	Ready
	M32	Error
CHANNEL	C1	Channel 1 A/D module
	C2	Channel 2 A/D module
	C12	Composite channel 1 & 2 display.
EOI, HOLD OFF	K0	EOI and hold off on X enabled
	K1	EOI disabled, hold off on X enabled
	K2	EOI enabled, hold off on X disabled
	K3	Both EOI and hold off on X disabled
TERMINATOR	Ym	Program m as single terminator
	Yn,m	Program mn as dual terminator
	Y (DEL)	No terminator
SAVE*	L1	Save setup 1
	L2	Save setup 2
	L4,m	Calibrate using value m ($-200 \leq m \leq +200$V)
	L5	Store cal in NVRAM
RECALL	A0	Recall factory setup
	A1	Recall setup 1
	A2	Recall setup 2
	A3	Recall setup 3
HIT BUTTON	Hn	Act as if button n (1-38) has been pressed
DISPLAY	Daaaa	Display ASCII characters aaaa (14 maximum)
	DX	Return to normal display mode
SELF TEST	J1	U1 self test byte =1: fail
EXECUTE	X	Execute other device-dependent commands.

*m parameter is in microvolts.

APPENDIX B
Data Formats

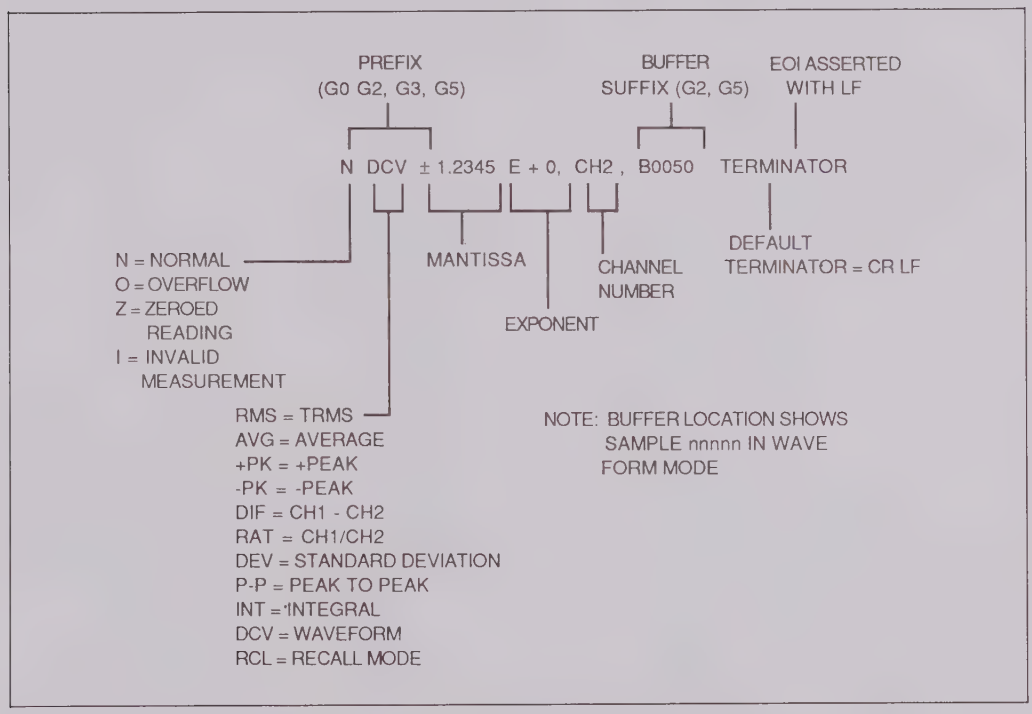


Figure B-1. ASCII Data Format (G0 to G5)

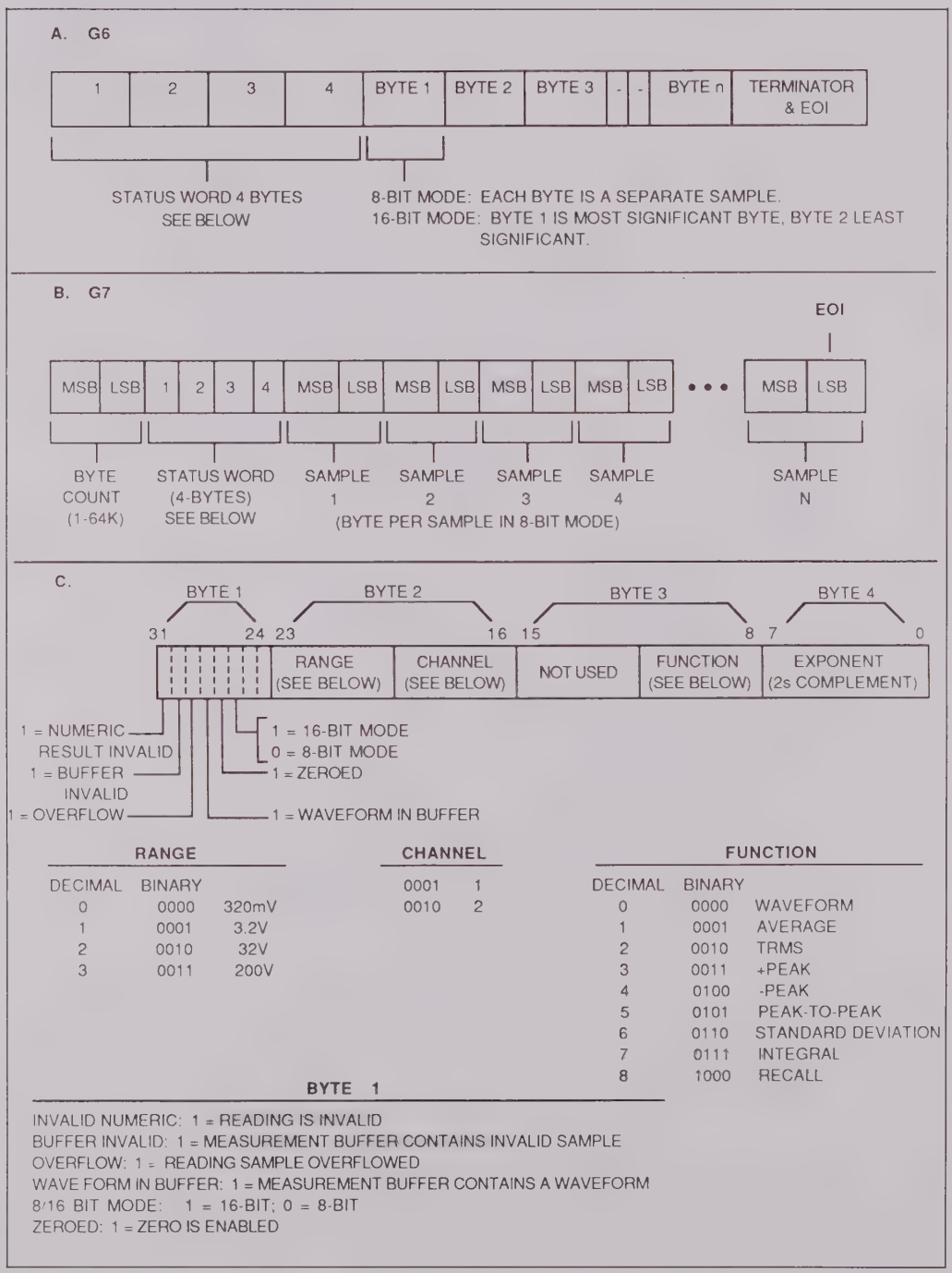


Figure B-2. Binary Data Formats

APPENDIX C
Status Words and SRQ Format

194 Fnn Rnn Tnn Pn Zn Kn Hnn In An Ln Qn Gn Jnn Cnn Mnnn Ynnnnnn TERMINATOR

MODEL NUMBER (194)

FUNCTION (Fnn)
00=WAVEFORM
01=AVERAGE*
02=TRMS
03=+PEAK
04=−PEAK
05=PEAK TO PEAK
06=STANDARD DEVIATION
07=INTEGRAL
20=CH1-CH2
21=CH1/CH2

RANGE (Rnn)
00=AUTO*
01=320mV
02=3.2V
03=32V
04=200V
12=CANCEL AUTO

TRIGGER (Tnn)
00=CONTINUOUS, TALK
01=SINGLE, TALK
02=CONTINUOUS, GET
03=SINGLE, GET
04=CONTINUOUS, X
05=SINGLE, X
06=CONTINUOUS, EXTERNAL
07=SINGLE, EXTERNAL
20=CONTINUOUS, +SLOPE
21=SINGLE, +SLOPE
22=CONTINUOUS, −SLOPE
23=SINGLE, −SLOPE
24=CONTINUOUS, OTHER CHANNEL
25=SINGLE, OTHER CHANNEL
26=CONTINUOUS, IMMEDIATE
27=SINGLE, IMMEDIATE
30=START PLOTTING
31=STOP PLOTTING

FILTER (Pn)
0=OFF*
1=500kHz
2=50kHz

ZERO (Zn)
0=DISABLED*
1=ENABLED
2=SAVE MEAS
3=USE VAL.
4=Z1 + Z2
5=Z1 + Z3

EOI, BUS HOLD OFF (Kn)
0=EOI, HOLD OFF*
1=NO EOI, HOLD OFF
2=EOI, NO HOLD OFF
3=NO EOI, NO HOLD OFF

BUTTON PRESSED (Hnn)
00=NONE
01=ZERO
02=FILTER
03=TRIGGER
04=SGL/CONT
05=SOURCE
06=DELAY
07=SLOPE
08=LEVEL
09=SCAN
10=CPLG
11=SAMPLES
12=RATE
13=LOCAL
14=CHANNEL
15=AUTO
16=▲ ►
17=◄ ▼
18=SHIFT
19=RECALL
20=STATUS
21=ENTER
22=CANCEL
23=FREQ/TIME
24=7
25=4
26=1
27=8
28=5
29=2
30=9
31=6
32=3
33=0
34=±
35=•
36=MHz
37=kHz
38=Hz

INPUT COUPLING (In)
0=DC COUPLING*
1=AC COUPLING
2=GROUND COUPLING

RECALL SETUP (An)
0=FACTORY SETUP*
1=SETUP 1*
2=SETUP 2

SAVE (Ln)
1=SAVE SETUP 1*
2=SAVE SETUP 2
4=CALIBRATE USING VALUE
5=STORE CAL IN NVRAM

READING BUFFER (Qn)
0=DISABLED*
1=LINEAR BUFFER
2=CIRCULAR BUFFER, OVERWRITE

DATA FORMAT (Gn)
0=ASCII, PREFIX ON, SUFFIX OFF, 1RDG
1=ASCII, PREFIX OFF, SUFFIX OFF, 1RDG
2=ASCII, PREFIX ON, SUFFIX ON, 1RDG*
3=ASCII, PREFIX ON, SUFFIX OFF, n RDGS
4=ASCII, PREFIX OFF, SUFFIX OFF, n RDGS
5=ASCII, PREFIX ON, SUFFIX ON, n RDGS
6=BINARY, PREFIX OFF, SUFFIX OFF
7=BINARY, PREFIX OFF, SUFFIX OFF, BYTE COUNT

SELF TEST (Jnn)
CODE/MEANING
00=NO ERROR
01=SELFTEST COMPLETE, NO ERROR (GOES TO 00 ON U0 READ)
02=NOT USED
03=NOT USED
04=NOT USED
05=NOT USED
06=NOT USED
07=NOT USED
08=RAM ERROR ON MOTHERBOARD
09=ROM ERROR ON MOTHERBOARD
1X=A/D ERROR ON CH1 (SEE TABLE BELOW FOR X MEANING)
2X=A/D ERROR ON CH2 (SEE TABLE BELOW FOR X MEANING)

A/D ERROR CODES
0=NOISY A/D
1=NOT USED
2=INOPERATIVE CAL ADJUSTMENT
3=NOT USED
4=NOT USED
5=TIMEBASE ERROR
6=NOT USED
7=RAM ERROR

CHANNEL (Cnn)
01=CHANNEL 1*
02=CHANNEL 2
12=COMPOSITE CHANNEL

SRQ (Mnnn)
000=DISABLED*
001=OVERFLOW
002=DATA
004=HIT BUTTON
008=READING DONE
016=READY
032=ERROR

TERMINATOR (Ynnnnnn)
nnnnnn=Y(ASCII)
000000=NO TERMINATOR
013010=CR LF*

TERMINATOR
DEFAULT CR LF

***FACTORY DEFAULT**

Figure C-1. U0 Status Word Format

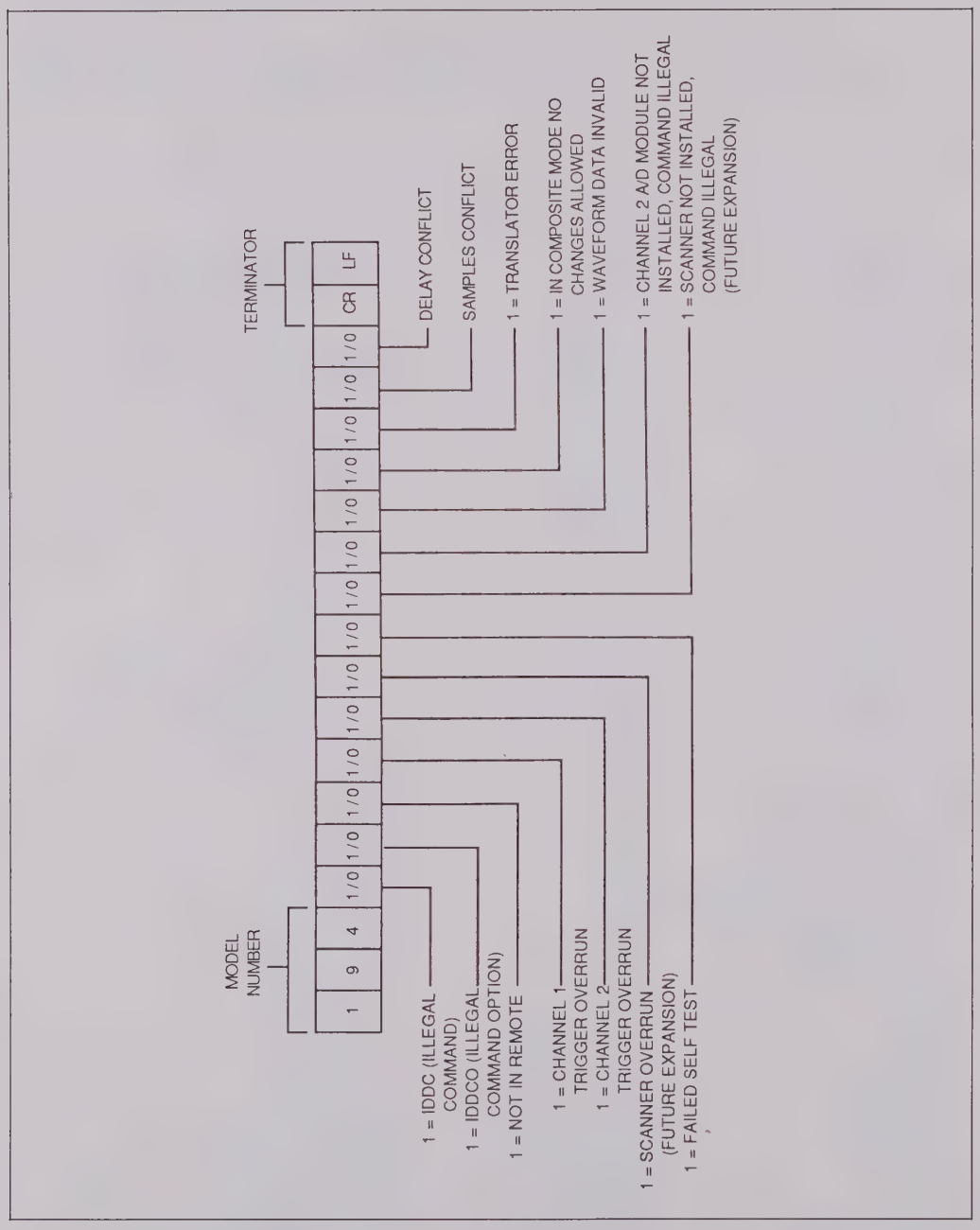

MODEL NUMBER
1
9
4
1/0
TERMINATOR
CR
LF
1 = IDDC (ILLEGAL COMMAND)
1 = IDDCO (ILLEGAL COMMAND OPTION)
1 = NOT IN REMOTE
1 = CHANNEL 1 TRIGGER OVERRUN
1 = CHANNEL 2 TRIGGER OVERRUN
1 = SCANNER OVERRUN (FUTURE EXPANSION)
1 = FAILED SELF TEST
DELAY CONFLICT
SAMPLES CONFLICT
1 = TRANSLATOR ERROR
1 = IN COMPOSITE MODE NO CHANGES ALLOWED
1 = WAVEFORM DATA INVALID
1 = CHANNEL 2 A/D MODULE NOT INSTALLED, COMMAND ILLEGAL
1 = SCANNER NOT INSTALLED, COMMAND ILLEGAL (FUTURE EXPANSION)

Figure C-2. U1 Status Word Format

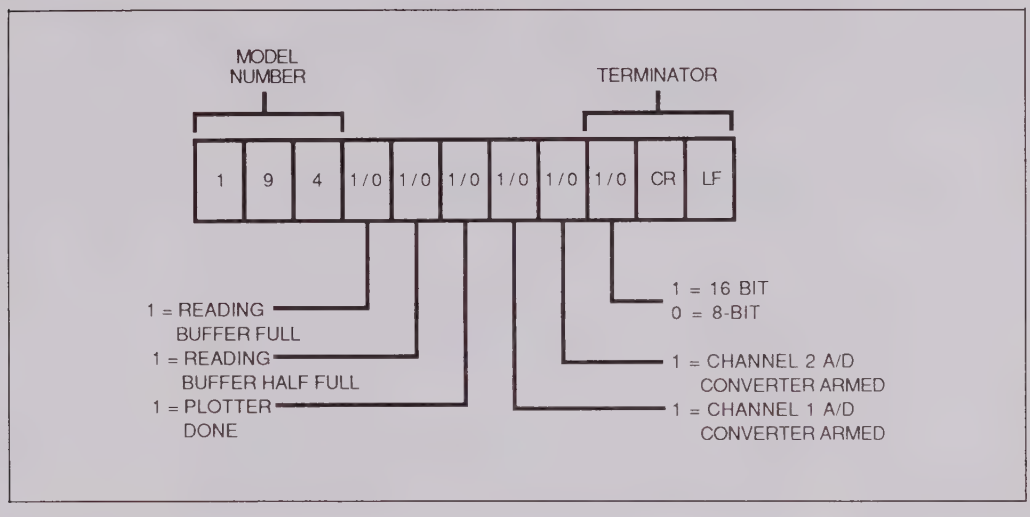


Figure C-3. U2 Status Word Format (Data Conditions)

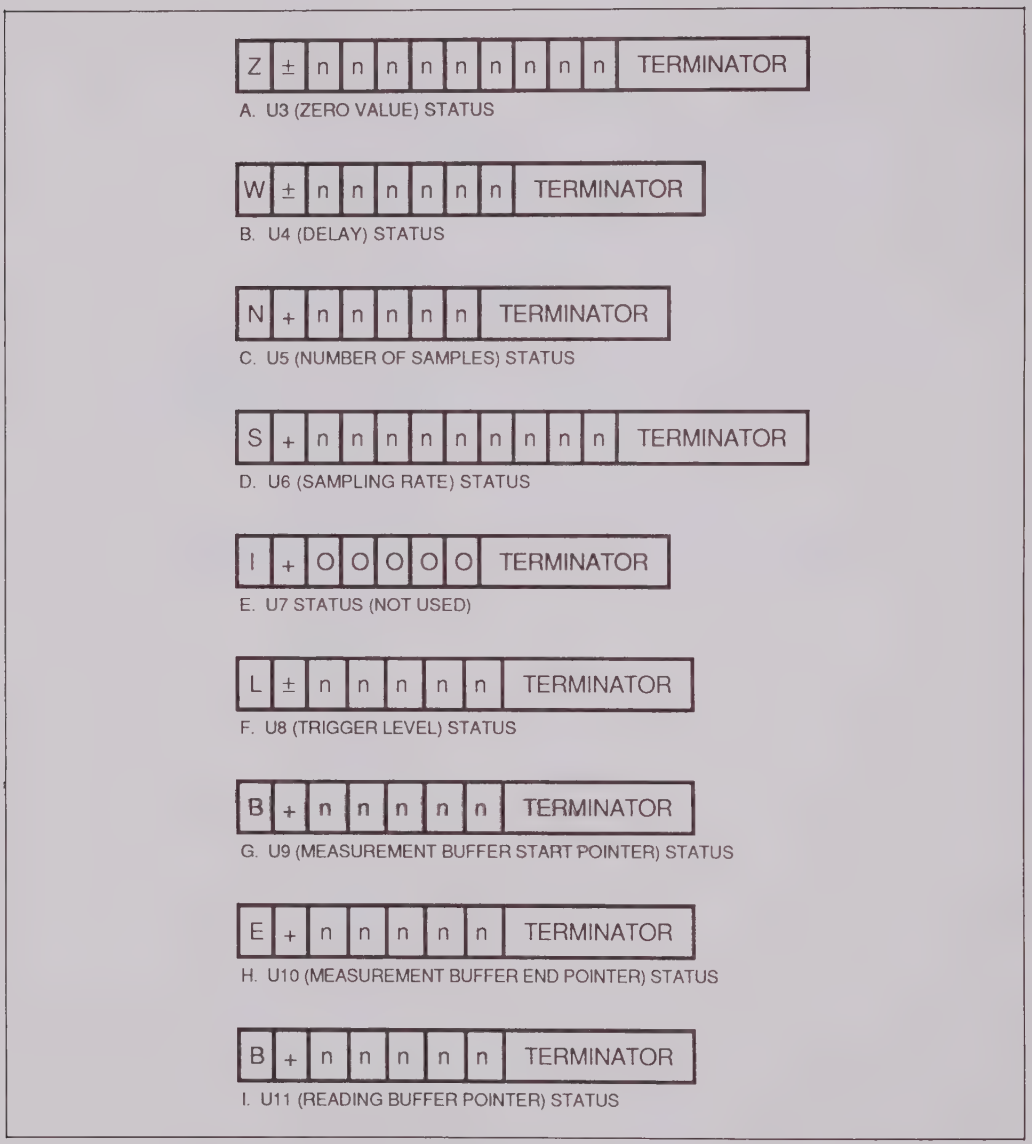

Z ± n n n n n n n n TERMINATOR
A. U3 (ZERO VALUE) STATUS
W ± n n n n n n TERMINATOR
B. U4 (DELAY) STATUS
N + n n n n n TERMINATOR
C. U5 (NUMBER OF SAMPLES) STATUS
S + n n n n n n n n n TERMINATOR
D. U6 (SAMPLING RATE) STATUS
I + O O O O O TERMINATOR
E. U7 STATUS (NOT USED)
L ± n n n n n TERMINATOR
F. U8 (TRIGGER LEVEL) STATUS
B + n n n n n TERMINATOR
G. U9 (MEASUREMENT BUFFER START POINTER) STATUS
E + n n n n n TERMINATOR
H. U10 (MEASUREMENT BUFFER END POINTER) STATUS
B + n n n n n TERMINATOR
I. U11 (READING BUFFER POINTER) STATUS

Figure C-4. U3-U11 Status Word Formats

APPENDIX D

Front Panel Modes and Equivalent IEEE-488 Commands

Front Panel Mode*	Equivalent IEEE-488 Command(s)
ZERO	Z4
(ZERO VAL)	Z5
FILTER	P
TRIGGER	T
(XY TRIG)	O0,O9
SGL/CONT	T
(XY MODE)	O2
SOURCE	T
(XY DATA)	O3
DELAY	W
(XY PAN)	O8
SLOPE	T20-T23
LEVEL	T20-T23
(XY ZOOM)	O7
CPLG	I
SAMPLES	N
RATE	S
LOCAL	GTL†
CHANNEL	C
AUTO	R0
Uprange	R1-R4
Downrange	R1-R4
SHIFT	None**
RECALL	F0,B
STATUS	U
(RESET)	A1
ENTER	None**
CANCEL	None**
FREQ/TIME	None**
Data Entry [0-9,.,+/-,MHz,kHz,Hz]	None**
(CH1÷CH2)	F21
(CH1-CH2)	F20
(OTHER)	None**
(WAVEFORM)	F0
(PK TO PK)	F5
(STD DEV)	F6
(INTEGRAL)	F7
(SETUP)	A,L
(AVG)	F1
(TRMS)	F2
(PEAK)	F3,F4

*Shifted modes shown in parentheses.

**Many of these modes are an intrinsic part of other commands. To emulate the front panel mode precisely, use the hit (H) command.

†Not a device-dependent command.

APPENDIX E

Data Transmission and NRFD Hold-off Times

Table E-1. Bus Data Transmission Times

Sampling Interval	Number of Samples	Math Function	Trigger to First Byte Out (msec)
166.7μsec	101	Average	47
166.7μsec	101	Peak-to-peak	40
166.7μsec	101	Integral	63
10μsec	1000	Average	122
10μsec	1	Average	22
1μsec	1000	Average	118
1μsec	1	Average	21
1μsec	1	Waveform	23
1μsec	1000	TRMS	208
1μsec	1000	+Peak	58
1μsec	1000	−Peak	58
1μsec	1000	Peak-to-peak	58
1μsec	1000	Standard Deviation	265
1μsec	1000	Integral	134

1. All measurements made in T1 (single, talk) mode.
2. All times are typical.

Table E-2. Typical NRFD Hold-off Times

Conditions	Hold-off (msec)
X TO X	15.7
F1 TO F5	132.8
F7 TO F0	132.8
R4 TO R0	163.2
S0,1 TO S0,1000	146.2
S1,1 TO S1,1000	210
N0,1 TO N0,1000	143.4
N1,.1 TO N1,.5	159.7
G1 TO G2	85.8
H1 TO H2	92.9
DNRFD TO D	77.8
P0 TO P2	212.7
T20,100 TO T23,−100	184.4
O2,0 TO O2,1	459.7
B0 TO B1,1B2,50	182.6
Q0 TO Q1	120.7
Z0 TO Z3,100	202.4
I0 TO I2	145.4
U0 TO U1	92.0
M0 TO M4	92.0
A0 TO A1	641.2
L1 TO L1	1760
C1 TO C2	135.6
K0 TO K1	128
J1 TO J1	13920

KEITHLEY

Keithley Instruments Division / Keithley Instruments, Inc.
28775 Aurora Road / Cleveland, Ohio 44139 / U.S.A. / (216) 248-0400 / Telex: 98-5469

WEST GERMANY: **Keithley Instruments GmbH** / Heiglhofstr. 5 / 8000 München 70 / 089-71002-0 / Telex: 52-12160 / Telefax: 089-7100259
GREAT BRITAIN: **Keithley Instruments, Ltd.** / 1 Boulton Road / Reading, Berkshire RG 2 ONL / 0734-861287// Telex: 847 047 / Telefax: 0734-863665
FRANCE: **Keithley Instruments SARL** / 3 Allee du 10 Rue Ambroise Croizat / B.P. 60 / 91121 Palaiseau/Cedex / 1-6-0115 155 / Telex: 600 933 / Telefax: 1-6-0117726
NETHERLANDS: **Keithley Instruments BV** / Avelingen West 49 / 4202 MS Gorinchem / P.O. Box 559 / 4200 AN Gorinchem / 01830-35333 / Telex: 24 684 / Telefax: 01830-30821
SWITZERLAND: **Keithley Instruments SA** / Kriesbachstr. 4 / 8600 Dübendorf / 01-821-9444 / Telex: 828 472 / Telefax: 0222-315366
AUSTRIA: **Keithley Instruments GesmbH** / Döblinger Haupstr. 32 / 1190 Wien / 0222-314-289 / Telex: 134 500 / Telefax: 0222-315366
ITALY: **Keithley Instruments SRL** / Viale S. Gimignano 4/A / 20146 Milano / 02-4120360 or 02-415640 / Telefax: 02-4121249